User's Manual for PZGui, v.8.0.xx

(*Pole/Zero Graphical-user-interface*)

a Matlab® Toolbox by Prof. Mark A. Hopkins, Ph.D.

Electrical & Microelectronic Engineering Department
Kate Gleason College of Engineering
Rochester Institute of Technology

July 2014

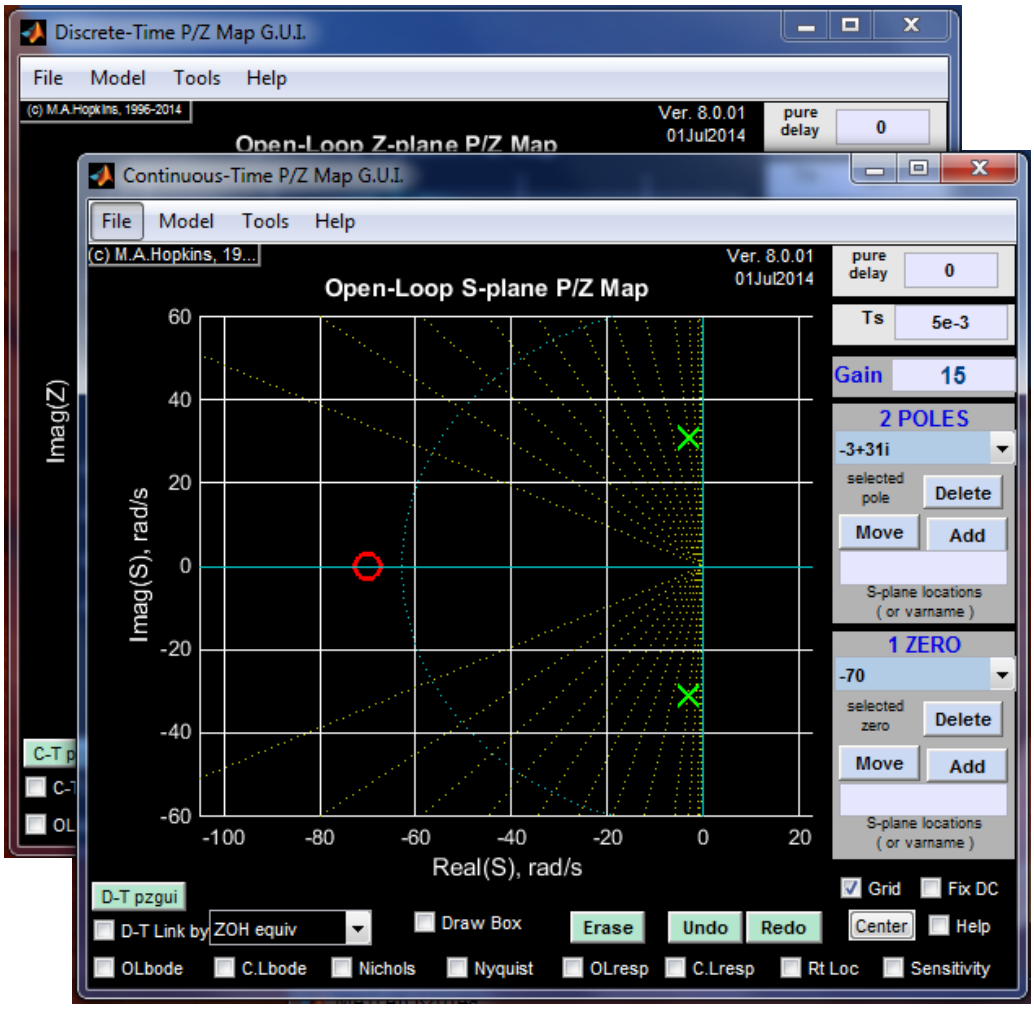


Table of Contents

Introduction

The toolbox **PZGui**, comprising over 65 m-files, is a Matlab® add-on tool that helps build understanding of the complicated relationships among the following:

- pole/zero maps
- frequency-response plots
  - open-loop Bode plots
  - closed-loop Bode plots
  - Nichols plot
  - Nyquist plot
  - Output sensitivity plots
- open-loop and closed-loop time responses
  - impulse, step, ramp, parabola, and sinusoidal inputs
- root locus
- the continuous-time domain, and the discrete-time domain

PZGui is useful in many situations. It was originally intended for the purpose of creating "bullet-proof" classroom demos. But over years of development, thanks to its increasingly advanced interactive and cross-linked graphics, it has become an excellent tool for understanding almost any aspect of single-input single-output transfer functions. It is suitable not only for students, but also for professionals trying to deepen their understanding of linear systems.

0. Installing and Running PZGui

Provided it is used only for educational purposes, **PZGui** is available ***free of cost*** via the author's home pages at RIT. Download the zip-file **pzgui80**xx**.zip** from

http://people.rit.edu/maheee/pzgui/Download_PZGUI_Matlab_toolbox.htm

Version 8.0.xx has been tested on several Matlab® versions, from 2008a through 2014b. It should be compatible with any version, from 2008a on up.

To ***install*** on your machine:

Unzip the files into a new subdirectory, *e.g.*, **\My Documents\Matlab\pzgui.**
From the Matlab® command-window menu, select **Set Path**.
In the user-interface window that comes up, click **Add Folder** and select that subdirectory.
You might want to ***save*** the new path before you close the set-path interface.

To ***run*** the program in Matlab®:

Once you have the **PZGui** subdirectory included in the path, at the Matlab® command prompt, simply type

```
>> pzgui
```

I. Overview of `PZGui`

This overview is written in the context of *continuous-time* transfer functions but, with only minor changes, it also applies to the discrete-time half of the **`PZGui`** tool (**`dpzgui`**).

`PZGui` is used to study single-input, single-output (SISO) linear systems. A system to be studied must be entered into the tool in "pole/zero/gain form". Several different ways to do that are described in section II, below.

In the main graphical user interface (GUI) figure, the system model is always represented as a **pole/zero map**, and an associated **transfer-function gain**. That corresponds to a Laplace-domain transfer function in zero/pole/gain factored form (also known as **zpk** form):

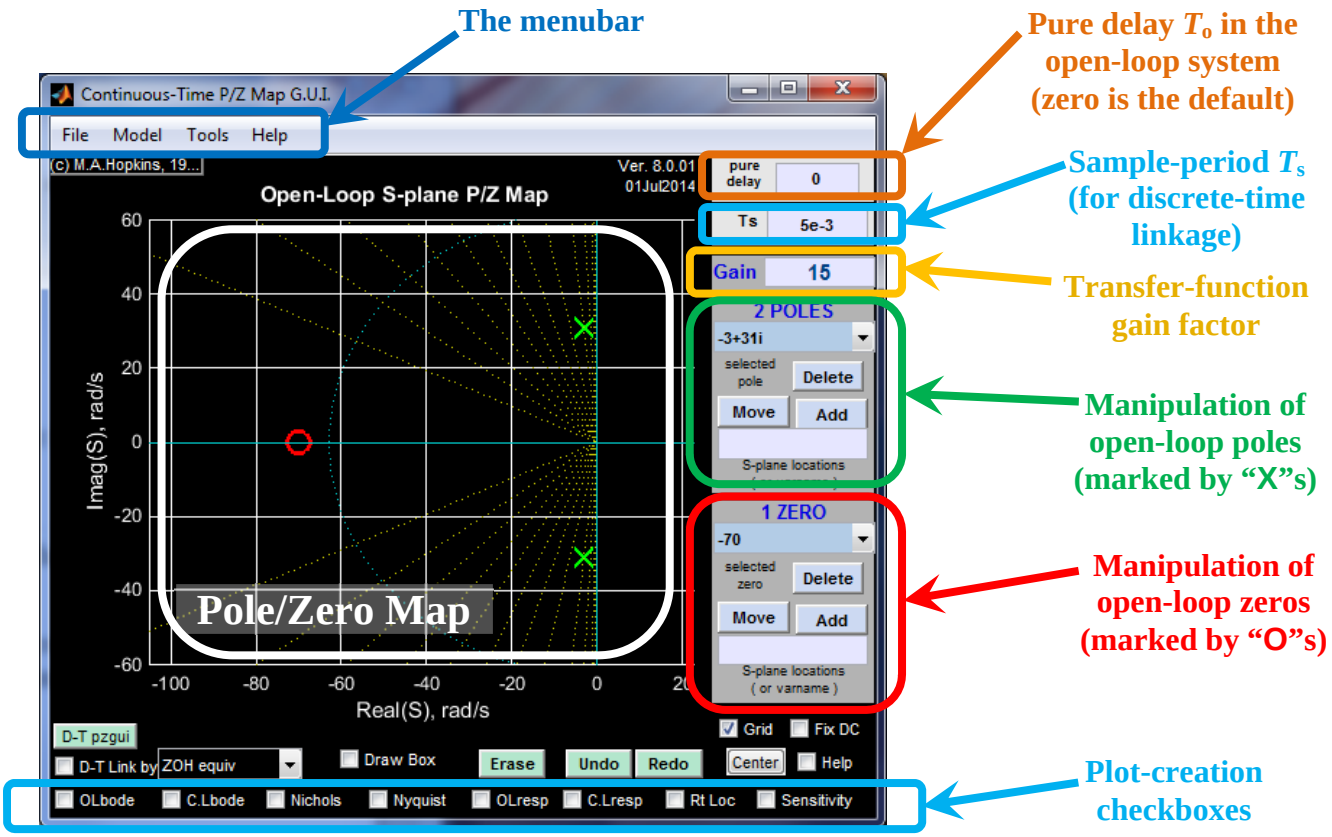
$$G(s) = \frac{K(s-\xi_1)(s-\xi_2)\cdots(s-\xi_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)}$$

Optionally, a nonzero **pure delay** can be included by specifying T_o the number of seconds of delay (or, in discrete-time, the number-of-samples of delay), so that the model corresponds to:

$$G(s) = e^{sT_o}\left[\frac{K(s-\xi_1)(s-\xi_2)\cdots(s-\xi_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)}\right]$$

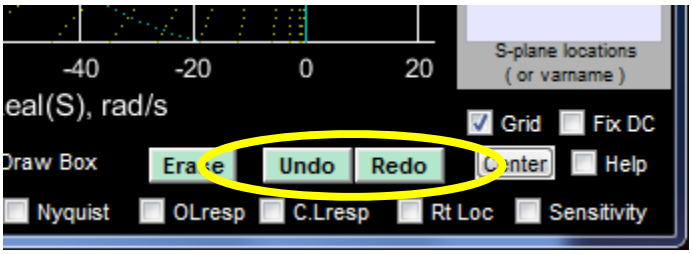
It is a ***crucial*** point that any model entered into this tool is **always** treated as an **open-loop system**. Moreover, the various closed-loop plots are always based on the assumption of standard output feedback (*i.e.*, the output is fed back and subtracted from the input) around this open-loop system.

The main interface figure is pictured below, with some of the interface areas highlighted.

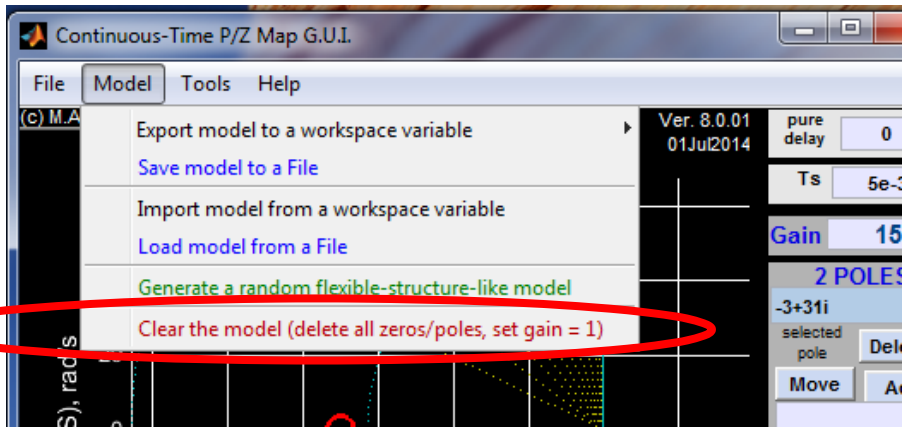


II. Entering a Model into `PZGui`

When you first start up **`PZGui`**, the main graphical user interface (GUI) figure will appear. Initially, there is a ***default second-order system*** defined, but it can easily be deleted, replaced, or modified. Any time a model is deleted, replaced or modified, the change can be "undone" using the "**Undo**" button. There is also a "**Redo**" button, to go back again.

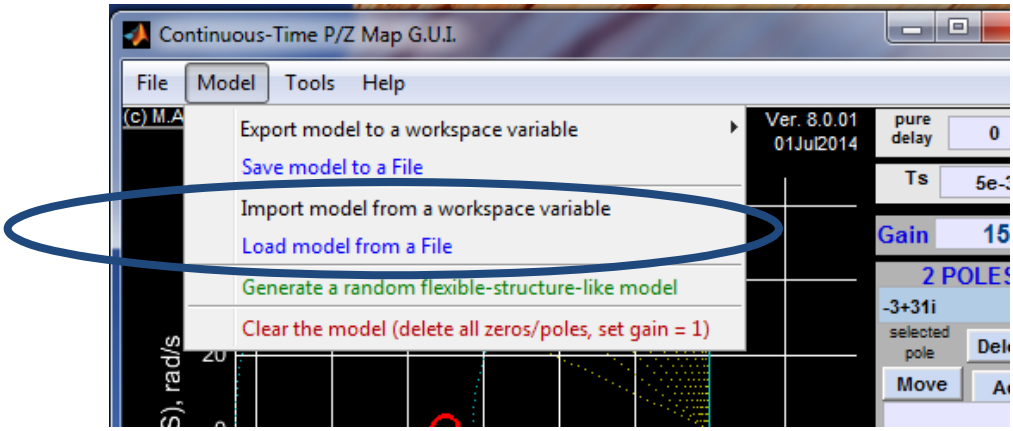


(1) Deleting the current model



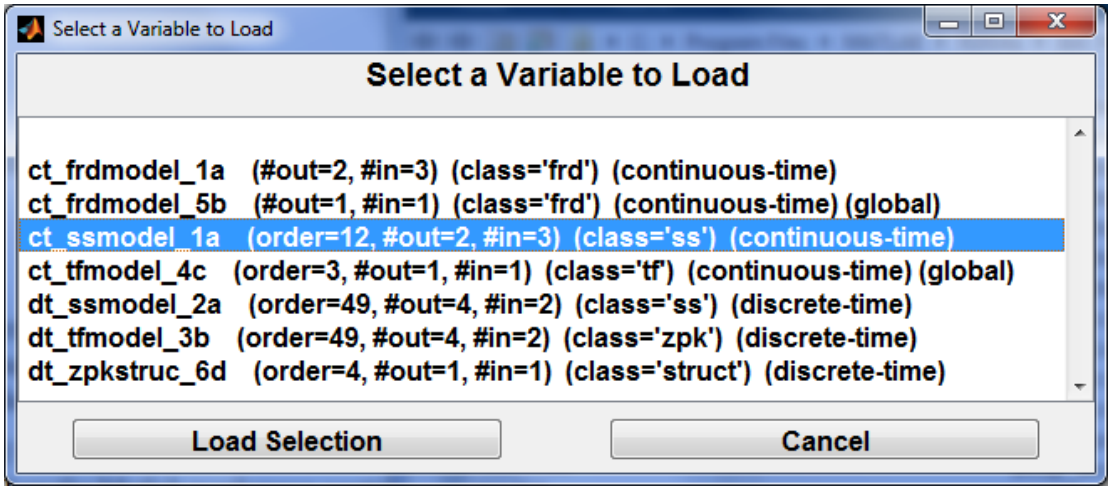
(2) Importing and loading models

The current model can be replaced by either one of the two selections highlighted below. Models can be imported from objects in the Matlab® workspace or loaded from **mat**-files. If the specified object is multi-input, multi-output (MIMO), you will be prompted to select *one of* the inputs and *one of* the outputs, so that **`PZGui`** can extract a *SISO submodel* from the specified MIMO object.



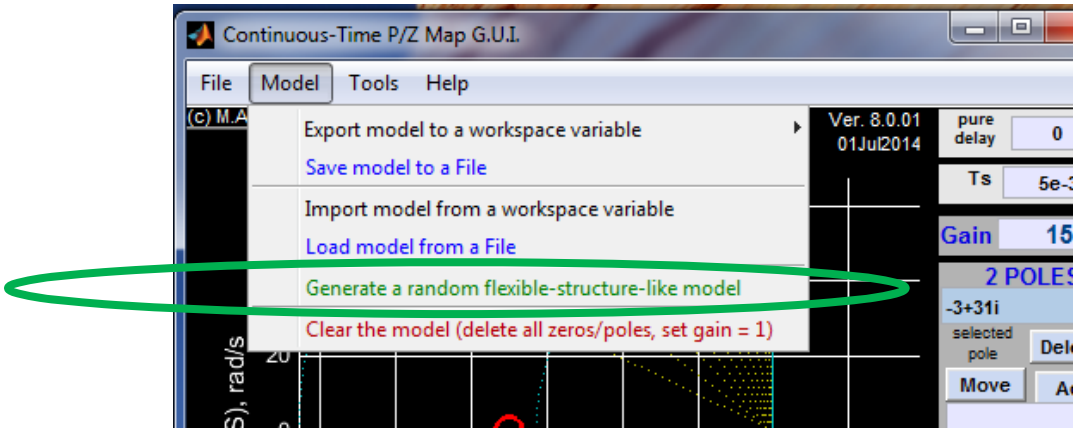
`PZGui` recognizes the Controls Toolbox standard objects "**`ss`**", "**`tf`**", and "**`zpk`**", and also can import frequency-response data from "**`frd`**" objects. It will also try to extract models from ***structure variables*** when they contain field names that *suggest* standard objects, such as a structure variable that contains the fields '**z**', '**p**', and '**k**', or one that contains the fields '**num**' and '**den**'.

Here is an example of the user-interface that appears when you select "Import model from a workspace variable", when the Matlab workspace contains any variables that **PZGui** recognizes as system models. In this example, there are several such variables.

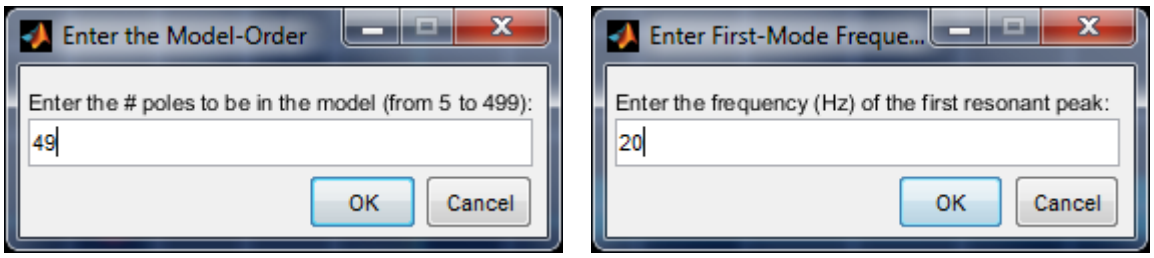


Note that if you load a "**frd**" (frequency-response data) object, it only shows up in the open-loop Bode plots, and has no effect on the transfer function that is loaded in the main interface figure. This capability is primarily intended to enable the comparison of frequency-response curves, such as during system identification.

There is yet another way to replace the current model, and that is to have **PZGui** generate a pseudo-random flexible-structure-like model, collocated to about 50x the first resonant-peak frequency.



This selection will bring up two successive windows in which to specify the model.



The subject of **saving a PZGui model** is covered in section **XIII**.

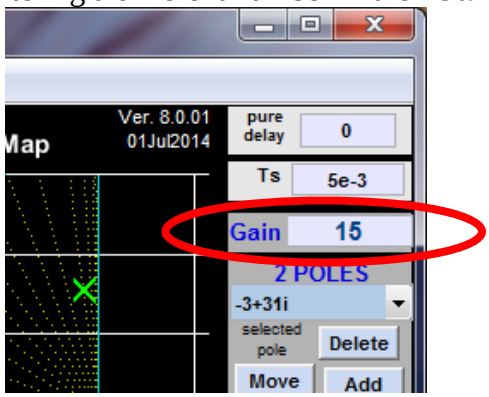
(3) Modifying an existing model

This is a very useful way to investigate how the frequency-response and time-response of a system depend upon the pole and zero locations, and the gain. There are several ways to do this, but they essentially come down to:

(a) changing the transfer-function gain
(b) deleting an existing zero or pole
(c) modifying an existing zero location or pole location
(d) adding new zeros or new poles

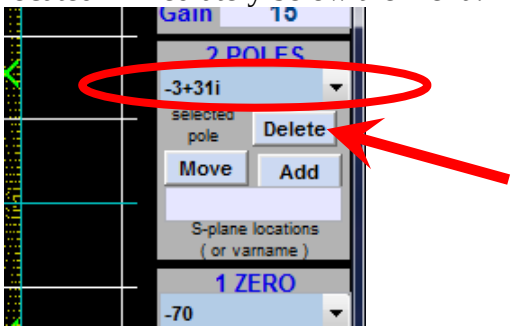
(a) Changing the transfer-function gain

This is done very easily, by entering a different number in the "**Gain**" window.



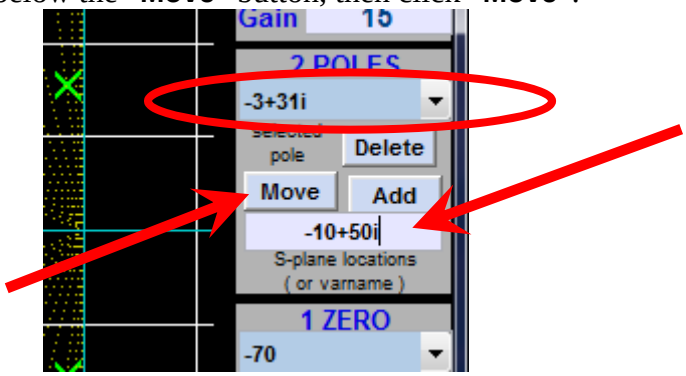
(b) Deleting an existing pole (similar for deleting a zero)

In the pole-selection dropdown menu, select the pole to delete.
Then click the "**Delete**" button, located immediately below the menu.



(c) Modifying an existing pole (similar for modifying a zero)

In the pole-selection dropdown menu, select the pole to modify.
Enter the new location just below the "**Move**" button, then click "**Move**".



(Modifying an existing pole, *continued …*)

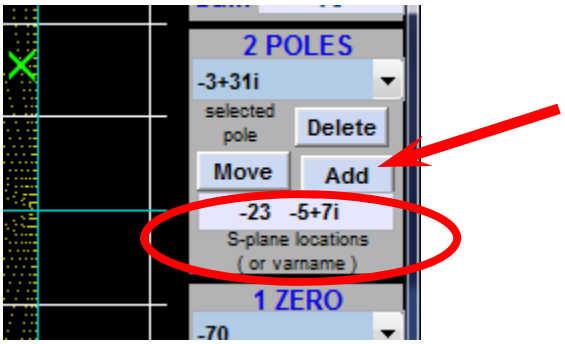
There is another simpler way to modify pole and zero locations. Using the mouse, you can **drag-and-drop** any pole or zero to a new location.

When you position the mouse cursor close to one of the poles or zeros in the pole/zero map, the cursor shape will change from the "magnifying glass" (*i.e.*, ready to zoom in or out) to the "hand". At that point if you press-and-hold either mouse-button, the nearest pole or zero will move with the mouse cursor (*i.e.*, it will be "dragged") until you release the mouse-button.

While you are dragging the pole (or zero), all of the plots that you have open (Bode, Nichols, Nyquist, time-response, *etc*.) will be continuously updated with "previews" of what they will look like if you release the button, dropping the pole or zero at that point. These previews are extremely useful in understanding the relationship of pole/zero location to frequency and time responses.

(d) Adding new poles (similar for adding zeros)

Enter numerical locations for the new poles just below the "**Add**" button, then click "**Add**".



Notice that you don't have to enter both poles of a complex-conjugate pair, because any non-real-valued poles (or zeros) are *always* added in conjugate pairs.

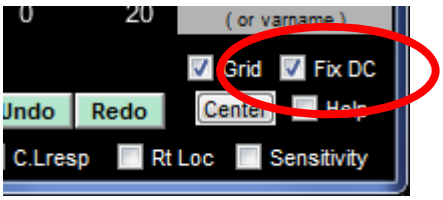
Also notice that you can *enter the **name of a variable** that is in the Matlab® workspace*, rather than entering numeric values. If that variable is a numeric vector, then **`PZGui`** will add poles at all of the locations specified in the elements of the vector.

III. An Important Option when Modifying a Model: *Fixing the DC Gain*

There is one major option that greatly affects what happens when a pole (or zero) is deleted, modified, or added. You can elect to have the dc gain held constant (fixed) when poles and zeros are changed. It is important to note that the ***dc gain*** is usually not the same as the ***transfer function gain***.

The transfer function gain is entered as described above, in section **II.3.a**, whereas the dc gain is the value of the transfer function evaluated at dc. For continuous-time systems, dc gain is $H(s)|_{s=0}$

For example, if a new pole is added at $s = -10$, then the transfer function gain must be increased by a factor of ten to maintain the same dc gain. This can be made automatic by selecting the checkbox labeled "**Fix DC**".



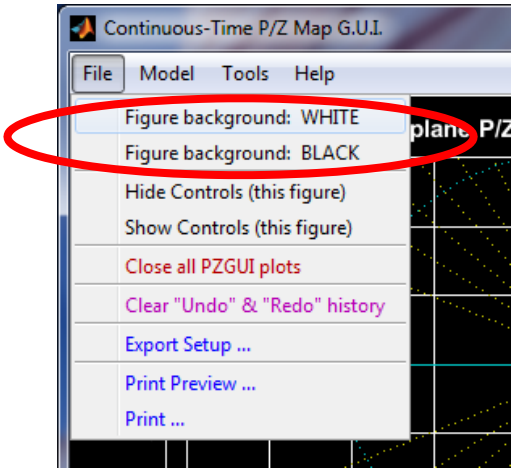
The usefulness of the "fixed dc gain" option is most apparent when a pole or zero is being dragged around. If there are frequency-response and/or time-response plots open, it is usually easier to interpret the effects of the changing location (as seen in the "previews") if the dc gain is not *also* changing.

IV. Other Options for Display and Format

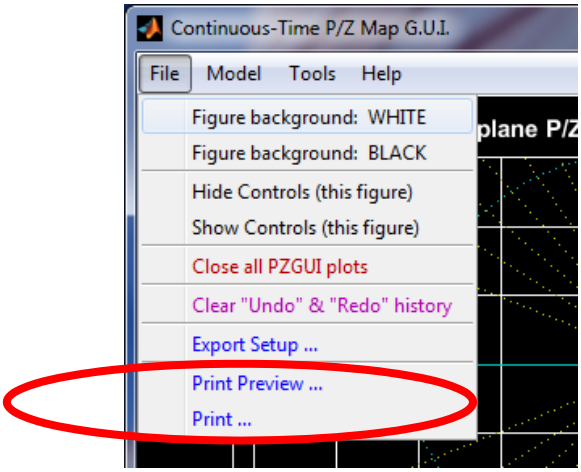
(1) Figure background color

There are two choices of background color: black or white. The default is black, because it is the best choice for overhead projection, and **`PZGui`** is, first and foremost, a classroom teaching tool.

The figure background color is easily changed, as shown here:

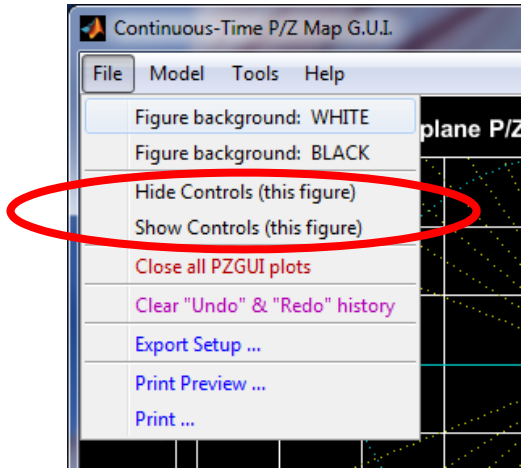


For ***printing*** a figure, a white background is usually the best choice, for two reasons. First, the amount of toner or ink required is significantly greater if the background is not white. Second, and more importantly, the resulting print usually *looks better* with a white background. For that reason, if you select either "**Print Preview ...**" or "**Print**", the background color will be changed temporarily to white, if it is not already white. When you finish with the printing process, it will be changed back to black, if it was black before.



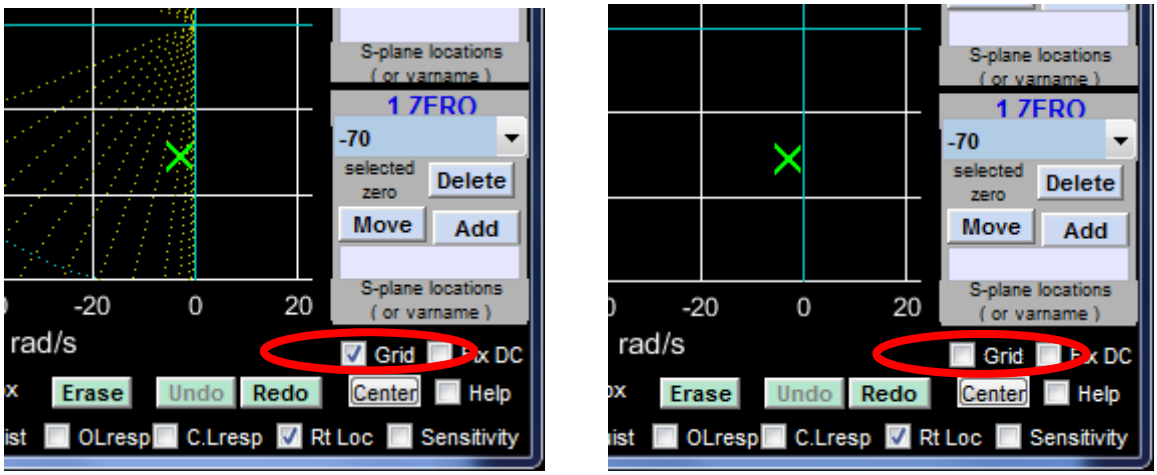
(2) Hiding the controls

There is an option to hide all the controls in a figure. This option is useful when having the controls appear in the figure distracts from the purpose of the figure. For the same reason, it is also useful when printing a figure.



(3) Hiding the pole/zero map grid-lines

There are pole/zero maps in the main interface figure and in the root locus plot. By default, the pole/zero maps have background "grids" that show the lines of constant damping-factor ζ, and lines of constant natural frequency ω_n. At times, these lines might distract from the purpose of the figure, so it is useful to be able to turn off their visibility. That is done by de-selecting the checkbox labeled "**Grid**".



There is a similar checkbox in the lower-right corner of the root locus plot.

(4) "Centering" the pole/zero maps

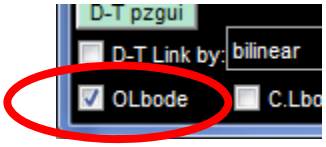
There are pole/zero maps in the main interface figure and in the root locus plot. Sometimes, in the process of zooming in and out, the scaling of the plot becomes less than useful. At such times, the pushbutton labeled "**Center**" will rescale the plot to a more useful region of the pole/zero map.

In the ***main interface figure***, the "**Center**" pushbutton appears just below the "**Grid**" checkbox, as can be seen in the figure directly above. By contrast, in the ***root-locus plot***, the "**Center**" pushbutton appears at the upper-right corner of the figure.

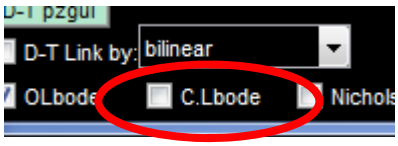
V. *Frequency-Response Plots*: Bode Plots and Output Sensitivity

On the **`PZGui`** main interface figure, there are five pushbuttons that create frequency-response plots.

(1) The open-loop Bode plots (magnitude vs. frequency, and phase vs. frequency)

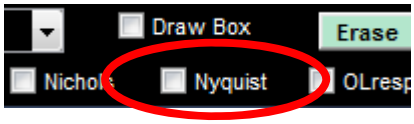


(2) The closed-loop Bode plots (magnitude vs. frequency, and phase vs. frequency)

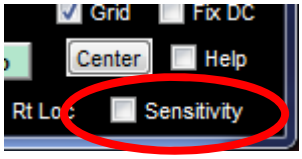


(3) The Nichols chart

(4) The Nyquist plot and the Nyquist contour



(5) The output-sensitivity function



***NOTE*: When the closed-loop system is *unstable*, the closed-loop Bode plots and output-sensitivity plot will not display any FRF information.**

The resulting plots are all interconnected and interactive. When you bring the mouse-cursor near the frequency-response line in any one of these plots, the corresponding point is highlighted in all of them, simultaneously. The highlighting displays the frequency at that point and, in the open-loop plots, also shows the magnitude and phase values at that frequency.

Moreover, the corresponding transfer-function evaluation point on the stability boundary is also highlighted in the pole/zero maps. For continuous-time, that will be a point on the $j\omega$-axis of the s-plane. For discrete-time, that is a point on the unit-circle $e^{j\omega T_s}$ in the z-plane.

There are some very useful user-interface controls that are located in the lower-left parts of these plots.

(a) x-axis units – either hertz or radians/second

(b) x-axis scaling – either log-scaled or linear-scaled

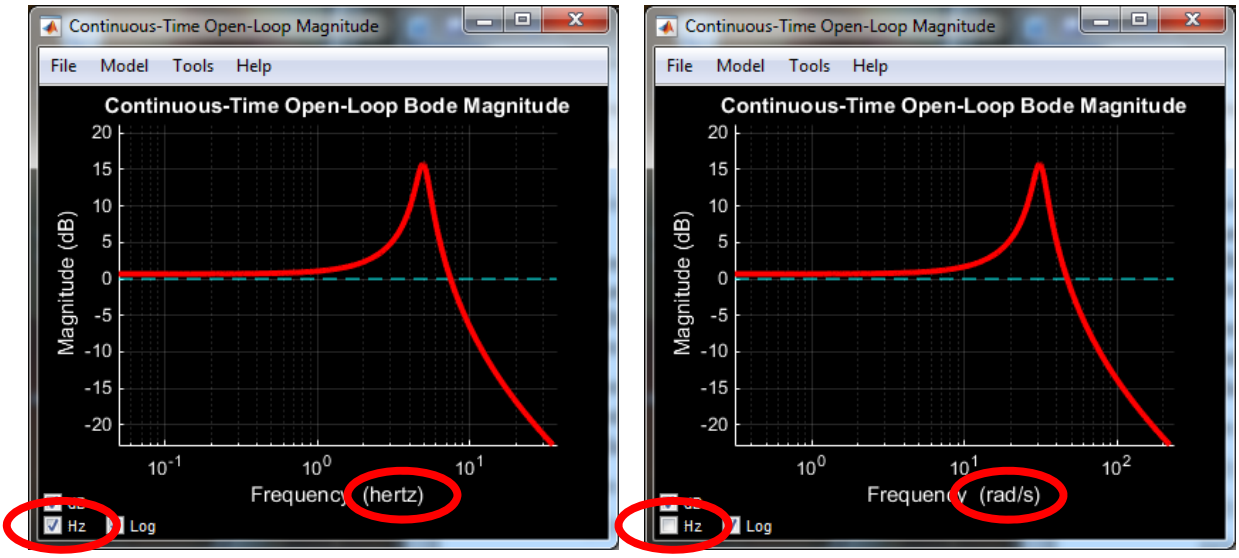
(c) y-axis units in the magnitude plots

(d) phase "unwrapping" in the phase plots

Whenever any of one these checkboxes is changed, the same change is applied to all of the other similar plots, both in the continuous-time and discrete-time domains. The next two pages show how these checkboxes affect the Bode plot format.

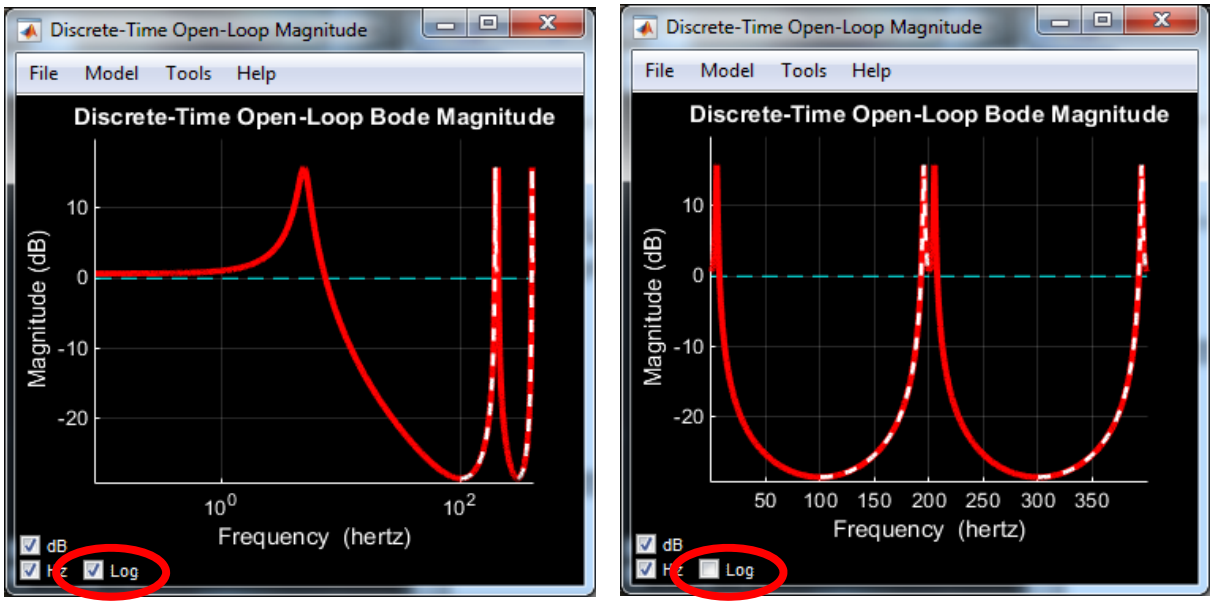
(a) x-axis units – either hertz or radians/second

Here is an example of changing frequency units from hertz to radians/second. When a Bode plot is created, the default is hertz, but it will be created in accordance with any pre-existing Bode plot.



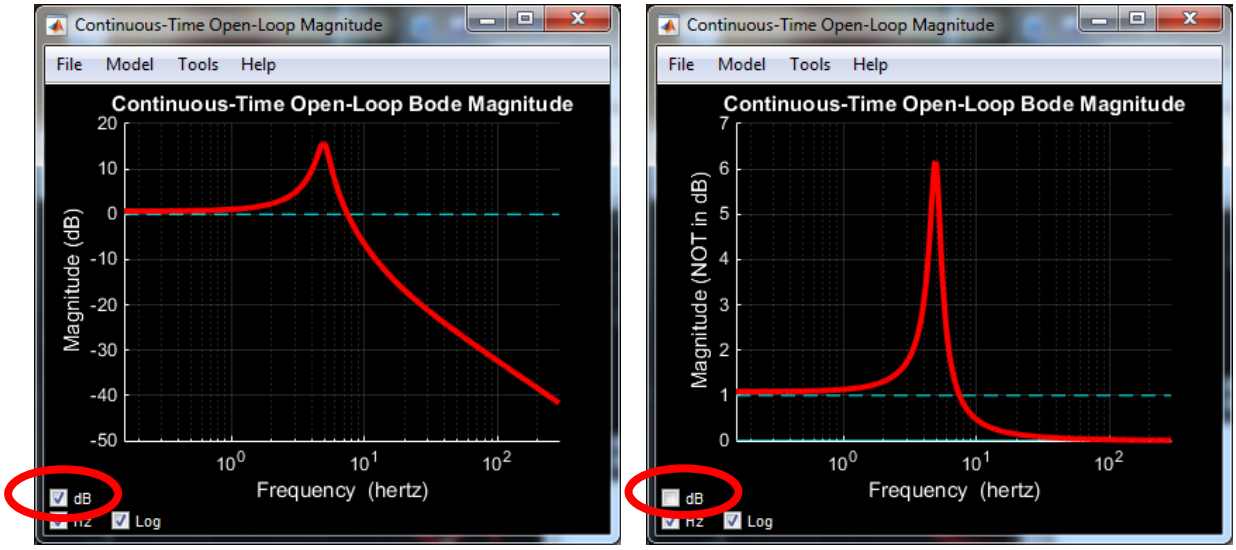
(b) x-axis scaling – either log-scaled or linear-scaled

Here is an example of changing x-axis scaling from ***log*** to ***linear***. This example is from a discrete-time plot, which is necessarily periodic at the sample frequency (in this case 200 Hz). When a Bode plot is created, the default is log-scaling, but it will be created in accordance with any pre-existing Bode plot.



(c) y-axis units in the magnitude plots

Here is an example of changing y-axis magnitude from units of decibels (dB) to raw magnitude. When a Bode magnitude plot is created, the default units are decibels, but it will be created in accordance with any pre-existing Bode magnitude plot.



(d) phase "unwrapping" in the phase plots

Here is an example of "unwrapping" the phase plot. When a phase plot is created, the default is "wrapped", but it will be created in accordance with any pre-existing phase plot.

Essentially, a phase plot that is wrapped is constrained to remain between $-180°$ and $+180°$, whereas an unwrapped plot is allowed to go beyond those limits, as in this example.

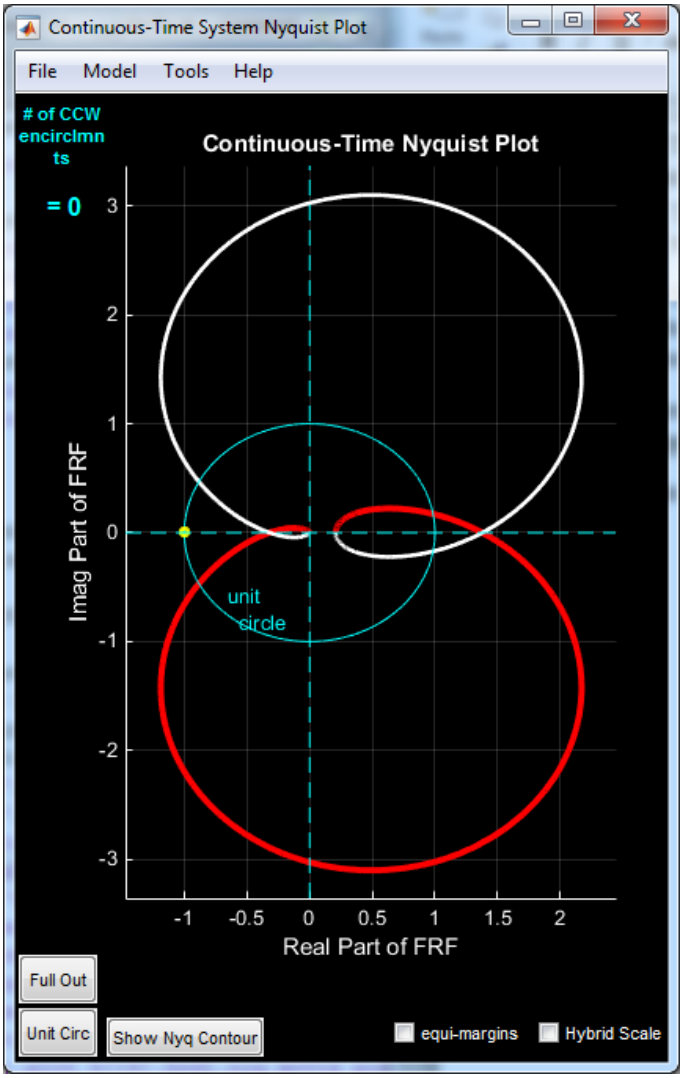
Wrapped **Unwrapped**

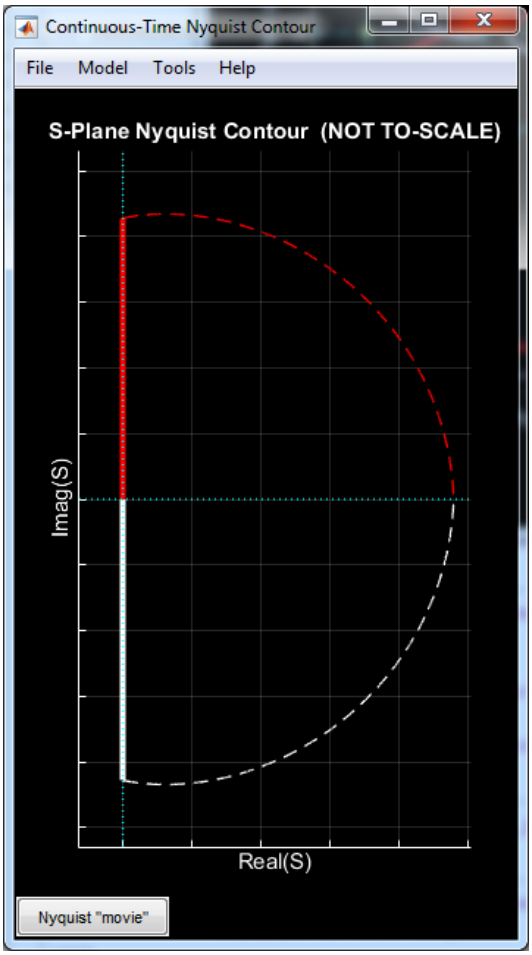
VI. *Frequency-Response Plots*: the Nyquist Contour and the Nyquist Plot

In **PZGui** the Nyquist contour and the resulting Nyquist plot have been developed into quite sophisticated tools. The following features are the reason.

- The contour is automatically adjusted for poles and zeros that lie exactly on the stability boundary. Specifically, the contour is automatically detoured around such points, into the unstable region, in a circle with infinitesimal radius (as specified by Nyquist).
  - Technically it isn't *necessary* to detour around ***zeros*** that are on the stability boundary, but it turns out that the resulting Nyquist plot is much more illuminating if you do.
- In **PZGui**, the Nyquist plot has a fairly unique nonlinear (logarithmic) scaling option that shrinks very large parts of the Nyquist plot and blows up very small parts, so they can be reasonably viewed simultaneously.
- There is a "**Nyquist movie**" capability, which shows how the Nyquist plot is generated, or mapped, as the transfer function is evaluated clockwise around the Nyquist contour.

Here is an example of these two plots.





Note the "**Nyquist movie**" checkbox in the lower-left corner of the Nyquist contour plot.

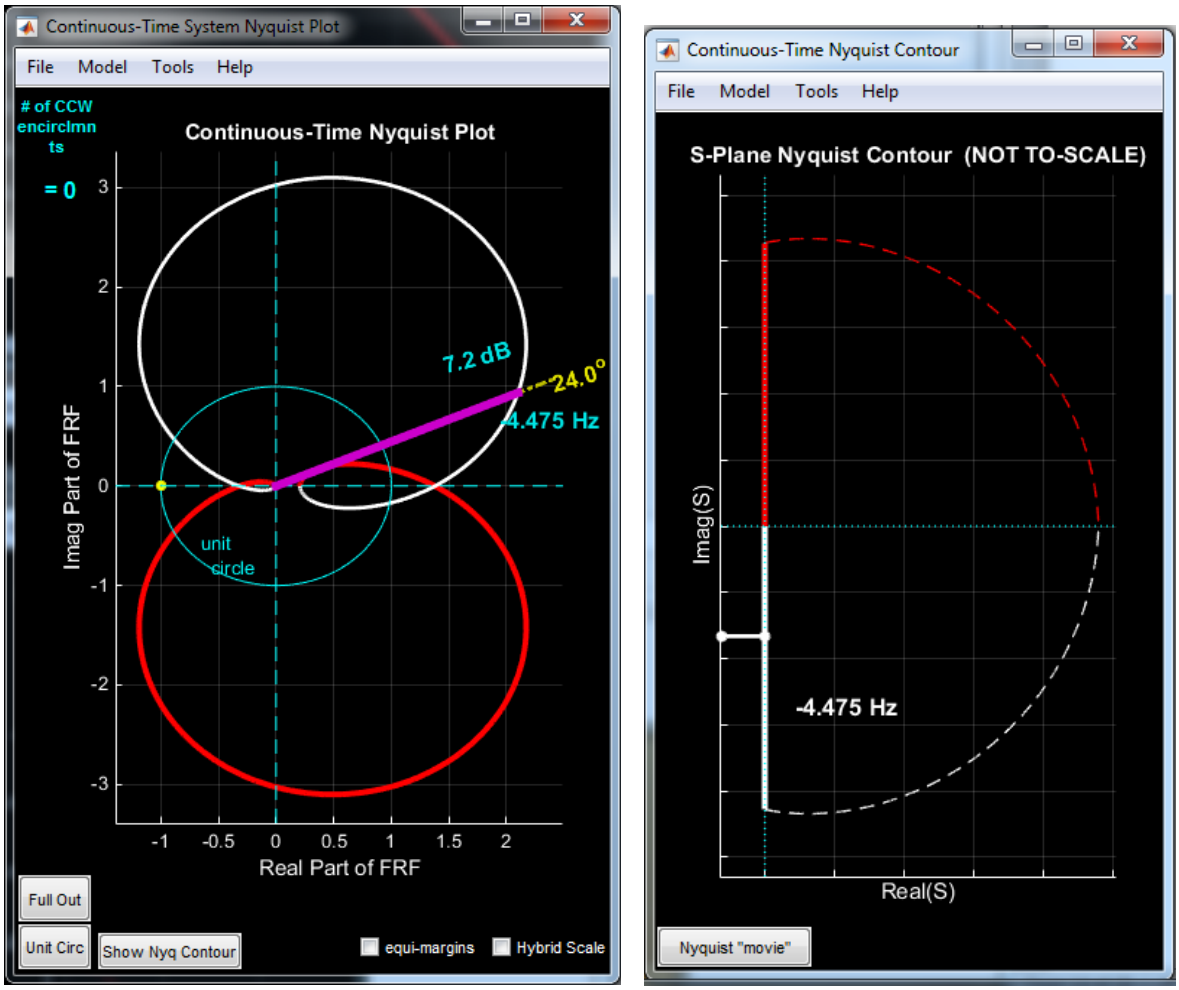
In the simplest *s*-plane Nyquist contour (*i.e.*, with no poles or zeros on the stability boundary), there are four parts of the contour,

(i) the positive frequency axis, also called the positive $j\omega$-axis
(ii) the negative frequency axis, also called the negative $j\omega$-axis
(iii) the upper half of the infinite-radius semicircle enclosing the right half-plane
(iv) the lower half of the infinite-radius semicircle enclosing the right half-plane

Each of these four parts is plotted in a different line style-and-color, and that is preserved in the mapping through the transfer function (*i.e.*, the Nyquist plot) from each part of the contour. That makes it easier to interpret the Nyquist plot.

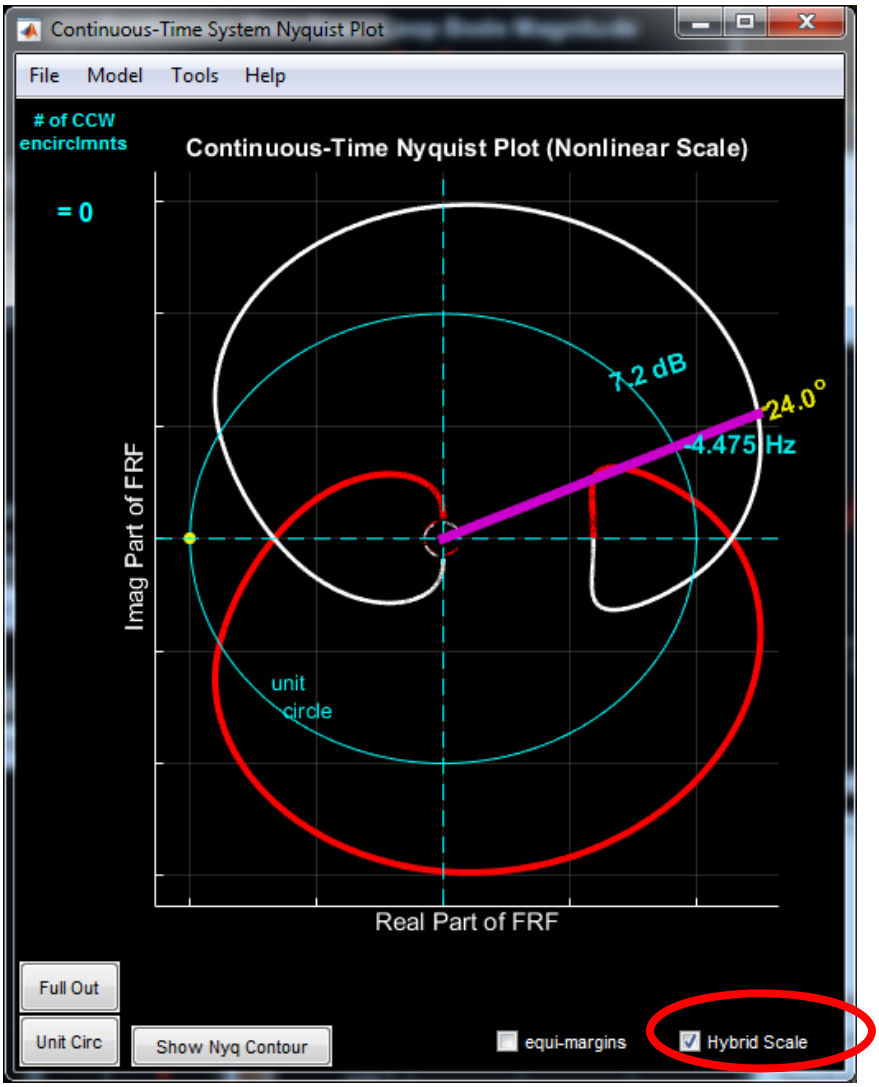
Also, if the mouse-cursor is brought close to any point on the Nyquist contour, that point is highlighted, and so is the corresponding mapped point on the Nyquist plot. The converse is also true: if you bring the mouse-cursor close to a point on the Nyquist plot, that point is highlighted and so is the corresponding source point on the Nyquist contour. These highlights include information about the frequency, magnitude, and phase.

Here is an example of highlighting, at frequency $\omega = -4.475$ Hz $= -28.12$ rad/s.



The "**Hybrid Scale**" checkbox turns on the nonlinear scaling option mentioned above. In the following figure, the same transfer function that generated the preceding figure was used, but hybrid scaling is

selected. Under this scaling it is possible to see, close to the origin, the dashed-line mapping from the infinite-radius part of the Nyquist contour.

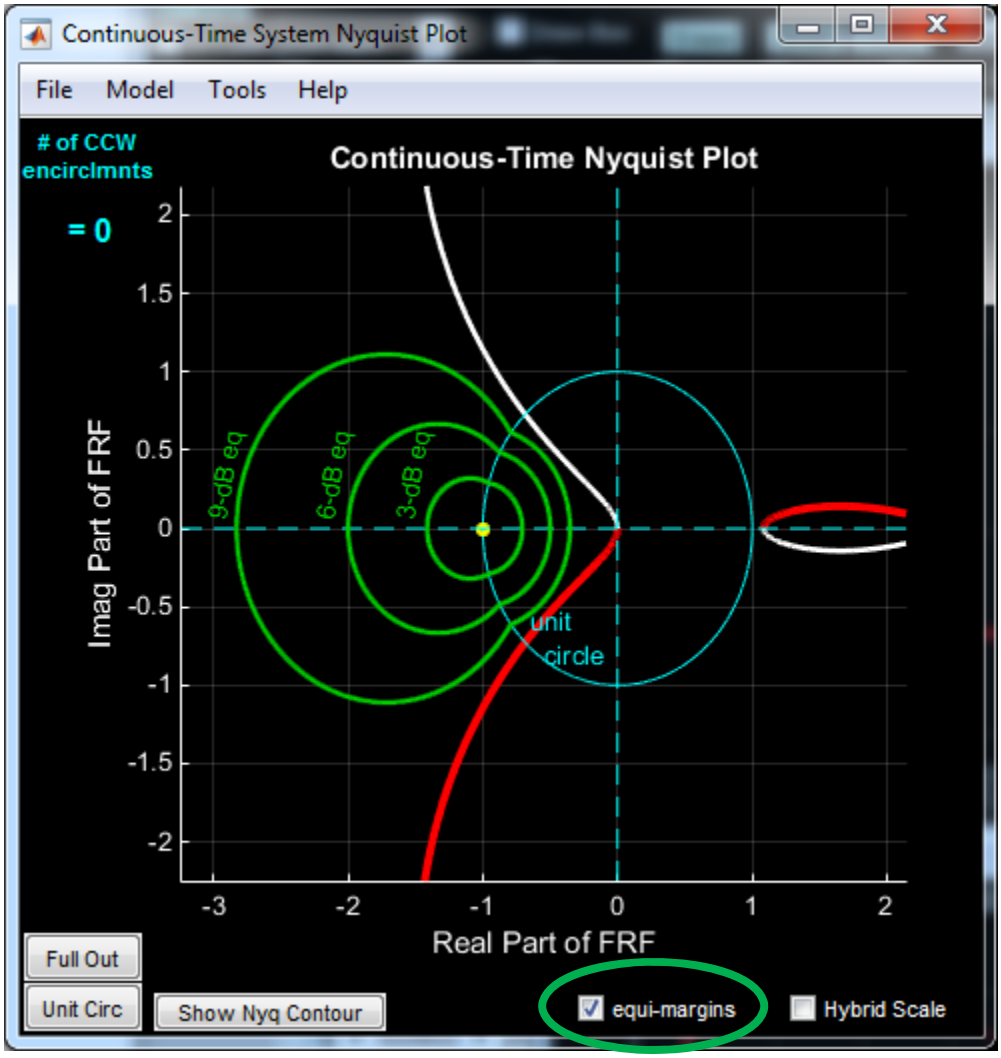


Only one part of the Nyquist contour is generally used in the other open-loop frequency-response plots, and that is the positive $j\omega$-axis, which is plotted as a solid red line (that same line style-and-color is used in the other plots). The exception is the Nichols plot, discussed in the next section, which has an option to show all of the other Nyquist plot data in the Nichols-chart style. That can be quite interesting and illuminating, particularly when the Nyquist plot has encirclements of the point -1.

The "Equi-Margin Grid"

The notion of a combined gain and phase margin can be useful. This has been called the *gain-phase margin*, and the idea is that it is important to consider the actual distance of the frequency-response plot from the point -1, not just in one "direction" (*i.e.*, gain direction or phase direction).

For that reason, the Nichols chart has an "**equi-margins**" checkbox. When this checkbox is selected, three equi-margin contours (3dB, 6dB, and 9dB) are shown as green lines, as seen in the next figure.



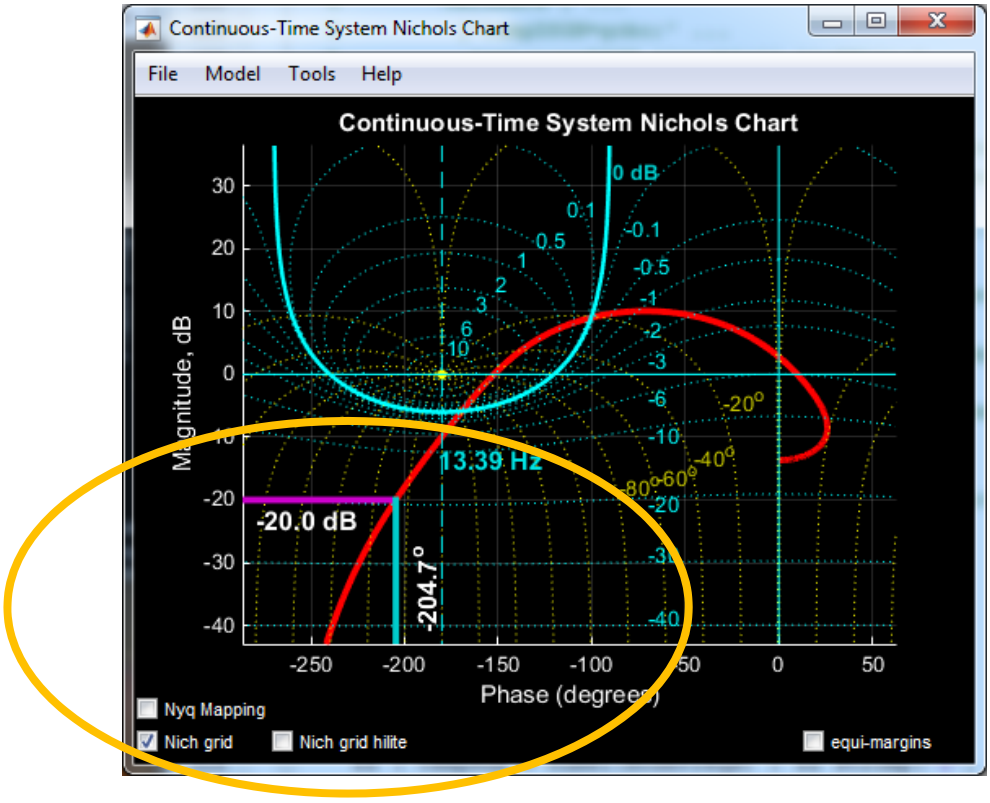
Lines of Constant Gain-Phase Margin

VII. *Frequency-Response Plots*: Nichols Plot

The Nichols plot, with its "Nichols grid", provides a visual link between the open-loop frequency-response and the closed-loop frequency-response. Because the Nichols plot is so important, there are some features of the **`PZGui`** Nichols chart that should be pointed out. These features make it easier to use a Nichols chart for design purposes.

(1) Frequency annotation

Because the Nichols chart is a plot of frequency-response magnitude (in dB) versus frequency-response phase (in degrees), the actual frequencies of the points along the plot cannot be directly determined. Frequency information is said to be "implicit". For that reason, when the mouse cursor is placed near the plot line, the frequency of the nearest point is displayed, as well as the magnitude and phase information at that point. Here is how that looks.

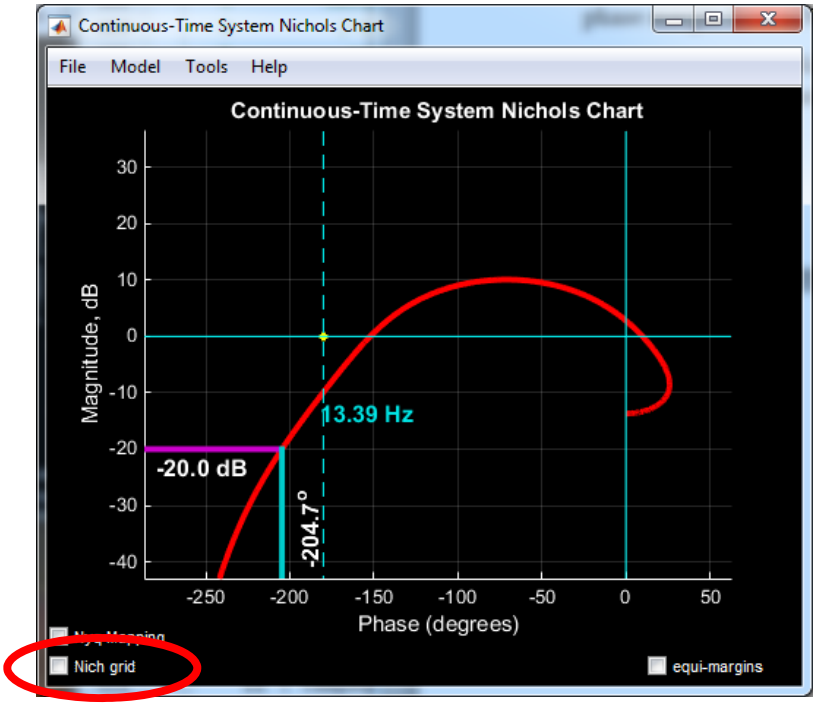


Because the cursor was placed near the point $(-204.7°, -20.0\text{ dB})$, that point is highlighted, and the frequency at that point (13.39 Hz) is also displayed nearby.

(2) The Nichols Grid and Highlight Option

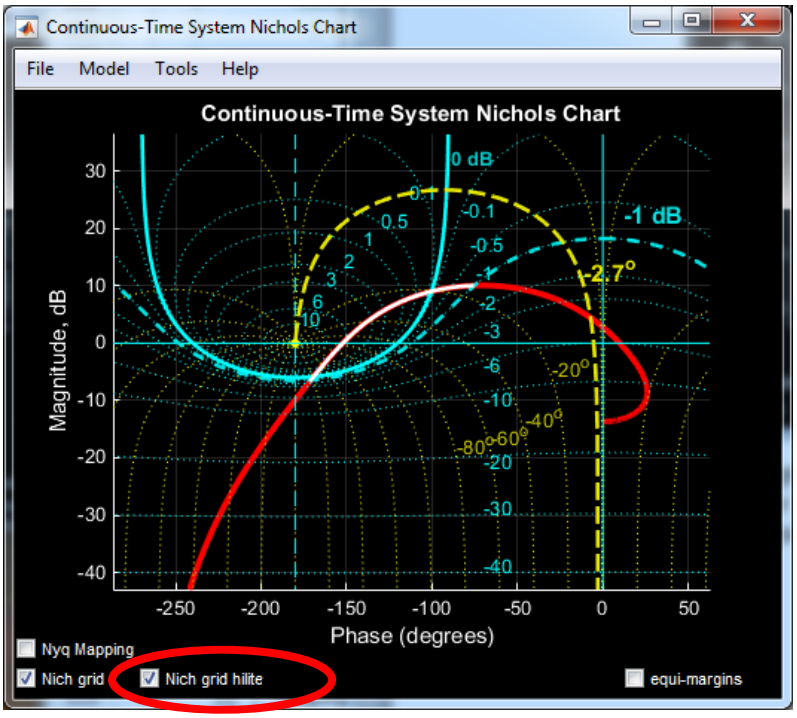
Although the Nichols grid is one of the main points of the Nichols chart, it is sometimes useful to be able to view the plot without it. For instance, when finding the closed-loop gain margin and phase margin from a Nichols plot, the Nichols grid is not used.

The visibility of the Nichols grid is controlled by the "**Nich grid**" checkbox. Here is the same plot viewed with the Nichols grid turned off.

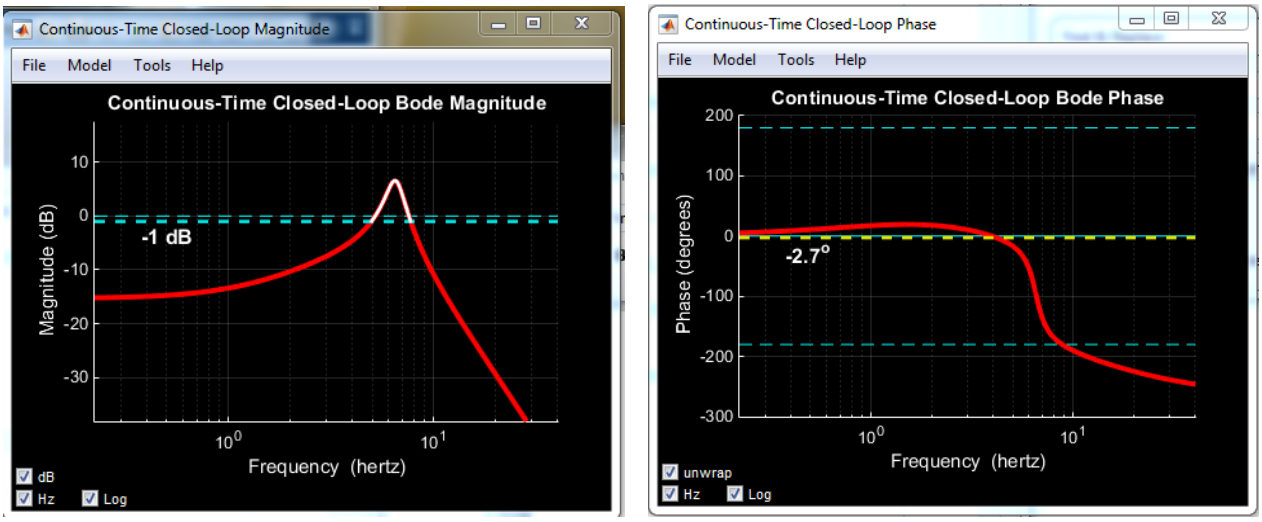


Because the Nichols grid is generally so important, you can elect to highlight the Nichols grid at the current mouse-cursor position. That feature is enabled by the "**Nich grid hilite**" checkbox, as shown in the next figure.

In the figure, the mouse-cursor was positioned at open-loop $(-20°, +17.5\text{ dB})$, which corresponds to closed-loop $(-2.7°, -1.0\text{ dB})$. The grid is highlighted by a cyan dashed line showing other points that map to -1.0 dB closed-loop magnitude, and a yellow dashed line showing other points that map to $-20°$ closed-loop phase. These two dashed lines intersect at the mouse-cursor position. As the mouse-cursor is moved, the dashed lines track its position, and display is updated.



Simultaneously, if the closed-loop Bode plots are open, that same magnitude and phase are highlighted in the closed-loop plots, as shown here.



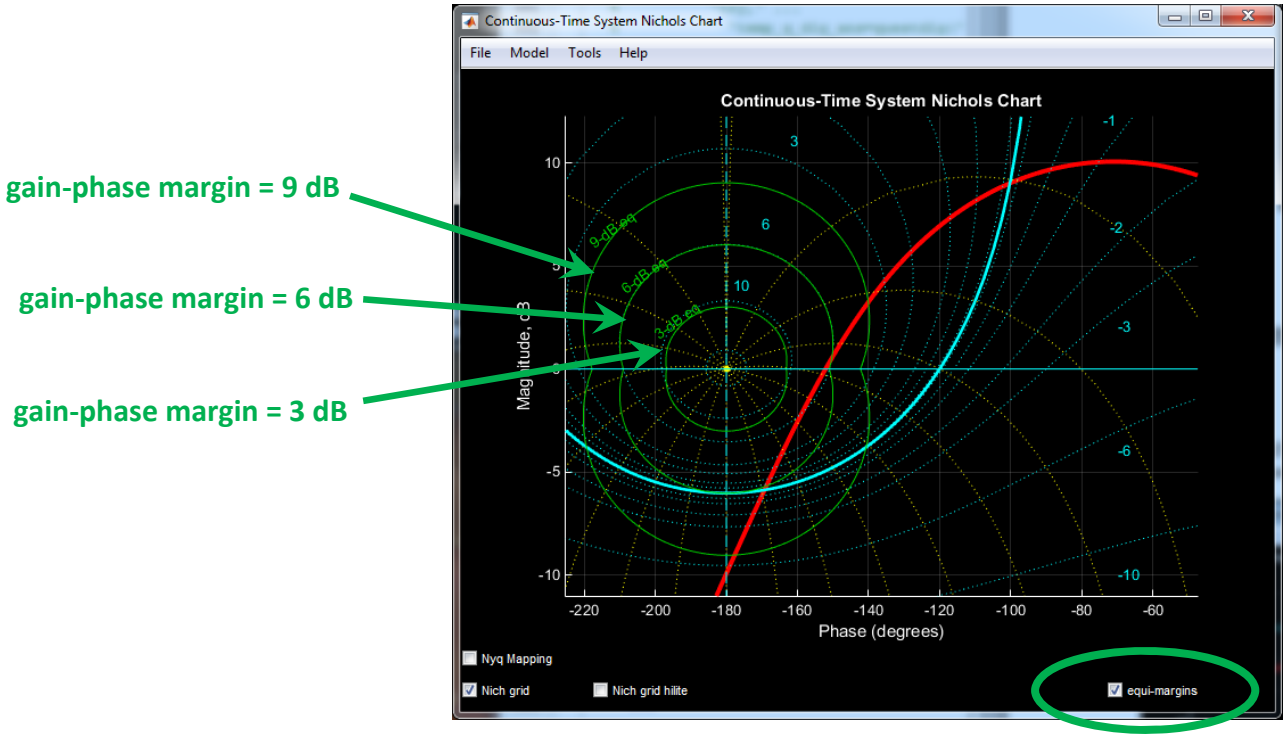
Notice that the frequencies at which the closed-loop magnitude plot exceeds -1.0 dB are highlighted (white) both in the Nichols plot and in the closed-loop magnitude plot.

This highlighting across multiple figures has proved to be very helpful to students trying to understand what the Nichols grid is all about.

(3) The "Equi-Margin Grid"

The notion of a combined gain and phase margin can be useful. This has been called the gain-phase margin, and the idea is that it is important to consider the actual distance of the frequency-response plot from the point -1, not just in one "direction" (*i.e.*, gain direction or phase direction).

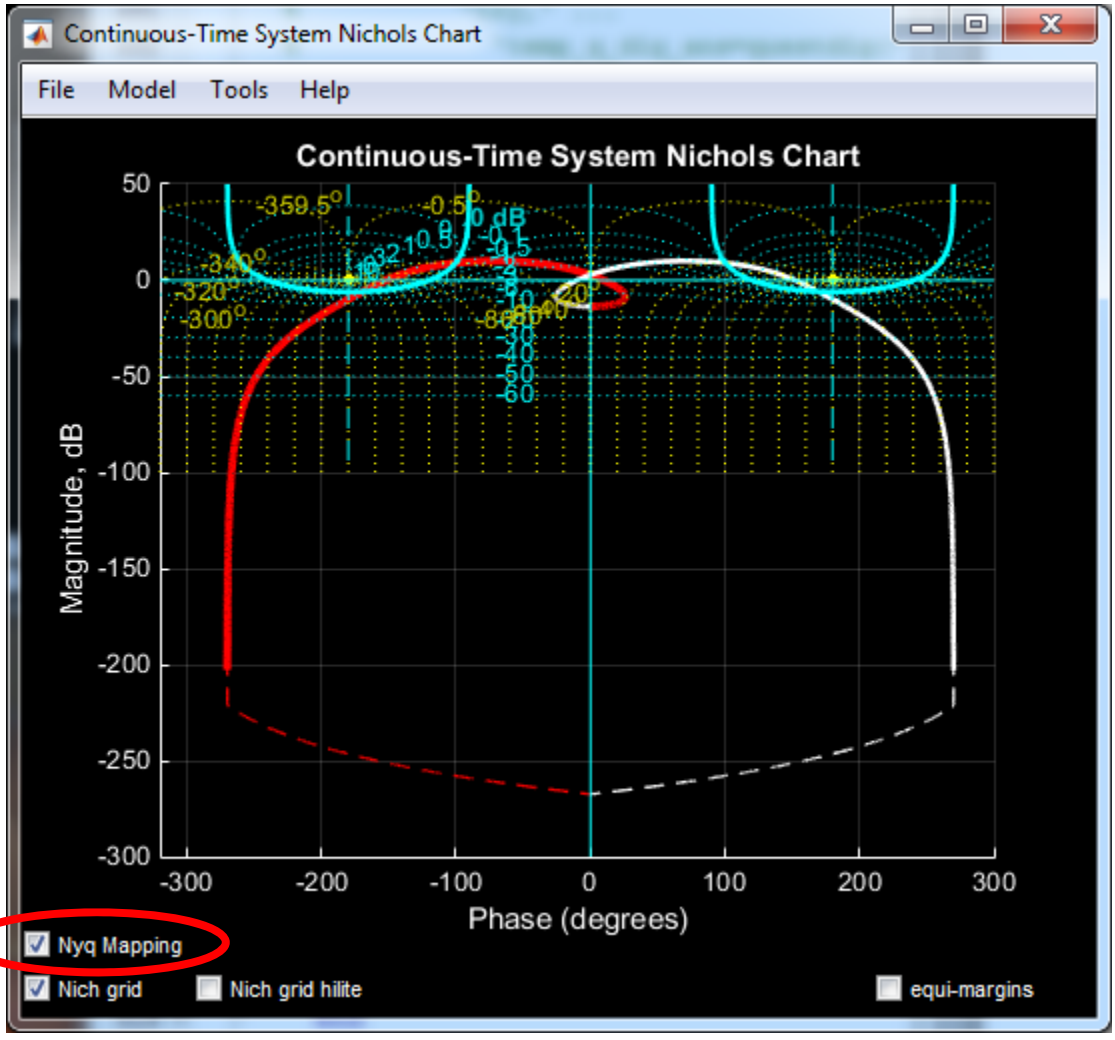
For that reason, the Nichols chart has an "**equi-margins**" checkbox.



There is one feature of the Nichols plot that presents a significant departure from the standard Nichols chart. The "**Nyq Mapping**" checkbox enables you to view the rest of the mapping from the Nyquist contour within a Nichols-chart context. This is quite interesting, particularly when there are encirclements of the point -1 in the Nyquist plot.

The non-Nichols points are indicated in white (negative $j\omega$-axis), and by the dashed lines which map from all the other parts of the Nyquist contour. The other parts are (1) any detours that must be made to avoid poles and zeros lying exactly on the stability boundary and (2) in continuous-time, the infinite-radius semicircle that completes the Nyquist contour's enclosure of the right-half of the *s*-plane.

When the "**Nyq Mapping**" option is selected, as seen in the figure below, all the points of the Nyquist plot are shown, as well as the Nichols plot. These two sets of data overlap along the solid red line, of course, because the Nyquist contour lies on the "positive" stability boundary (the positive $j\omega$-axis in the *s*-plane, and the upper half of the unit-circle in the *z*-plane).

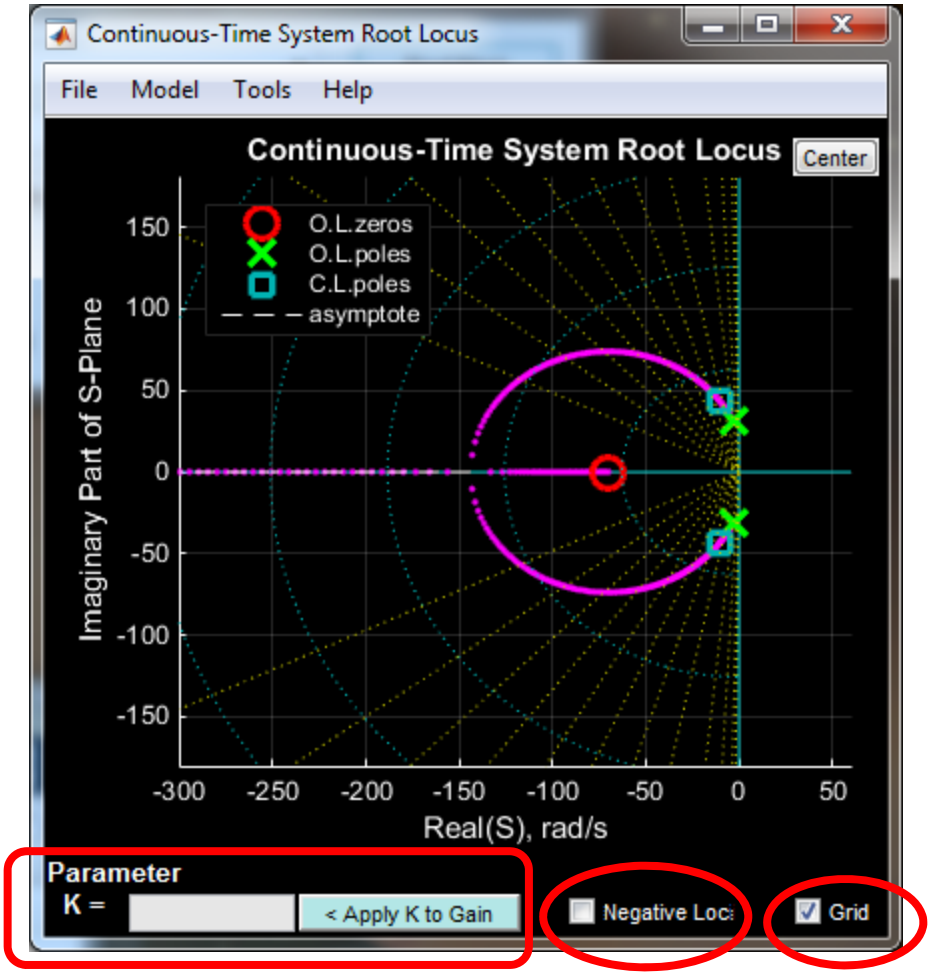


VIII. The Root-Locus Plot

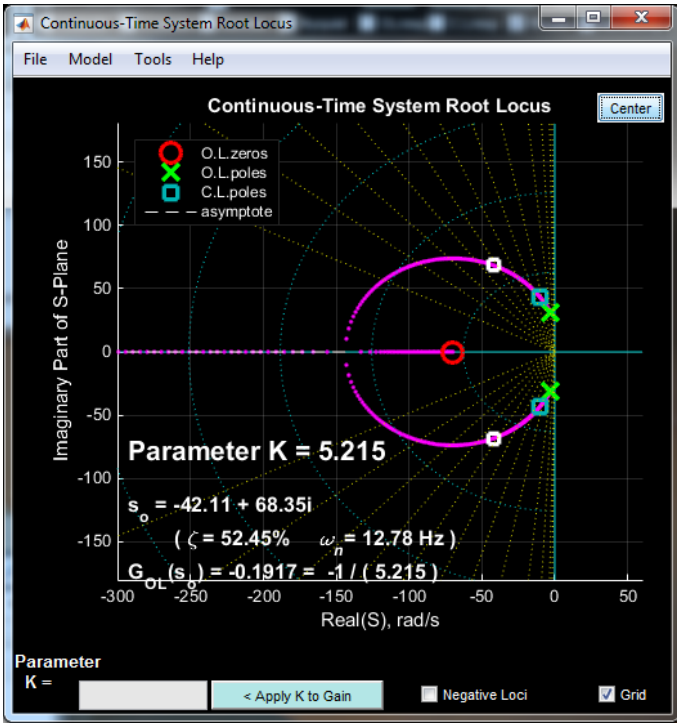
There are some useful and interesting features of the **`PZGui`** root-locus plot.

- Preview the effects of changing the gain by a factor "*K*"
  - ➢ Place the mouse-cursor near any branch of the root-locus
    - ▪ This previews the effect of a gain change in <u>every</u> open plot.
  - ➢ Click on any point along any branch of the root-locus
  - ➢ Enter the numerical value of *K* in the "**Parameter K =**" window
- Show the "*negative root-locus*" plot, *i.e.*, the plot that results from <u>negative</u> gains
- Show or hide the grid of constant damping-factor ζ and constant natural-frequency ω_n

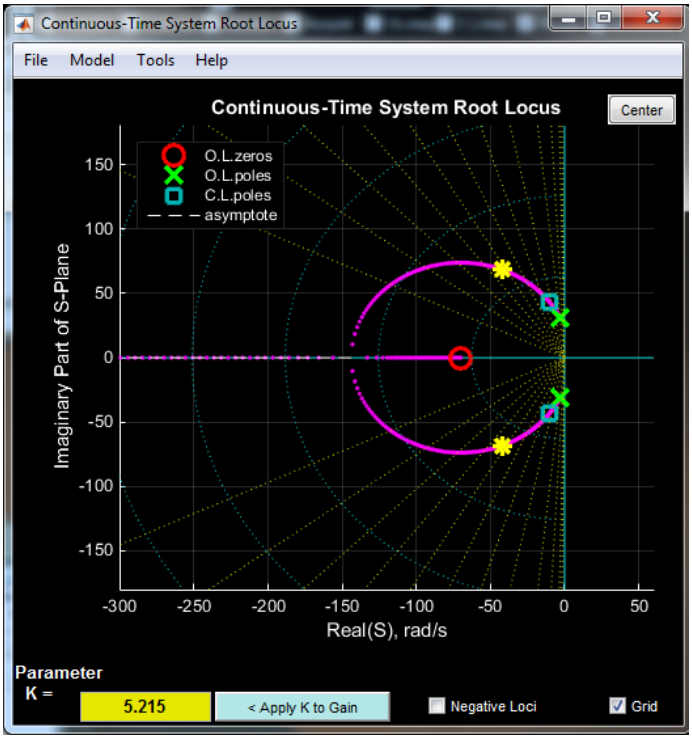
The following figure highlights the controls that are related to these features.



When the mouse-cursor is brought close to any branch of the root locus, the gain factor *K* that would put a closed-loop pole at that point is automatically displayed, as shown in the following figure. Notice that two other things happen. All of the closed-loop pole locations that would result from that additional gain factor are highlighted as small white squares, and the mouse-cursor is changed from its normal "magnifying glass" shape (*i.e.*, ready for zooming) to a small circle.



Whenever the mouse-cursor has that small circular shape, if you click the left mouse-button, the indicated gain will be copied into the "**Parameter K =**" window and the preview locations will be highlighted as yellow asterisks. For example, with the mouse in the same location as the previous figure, a left-click results in the following figure.

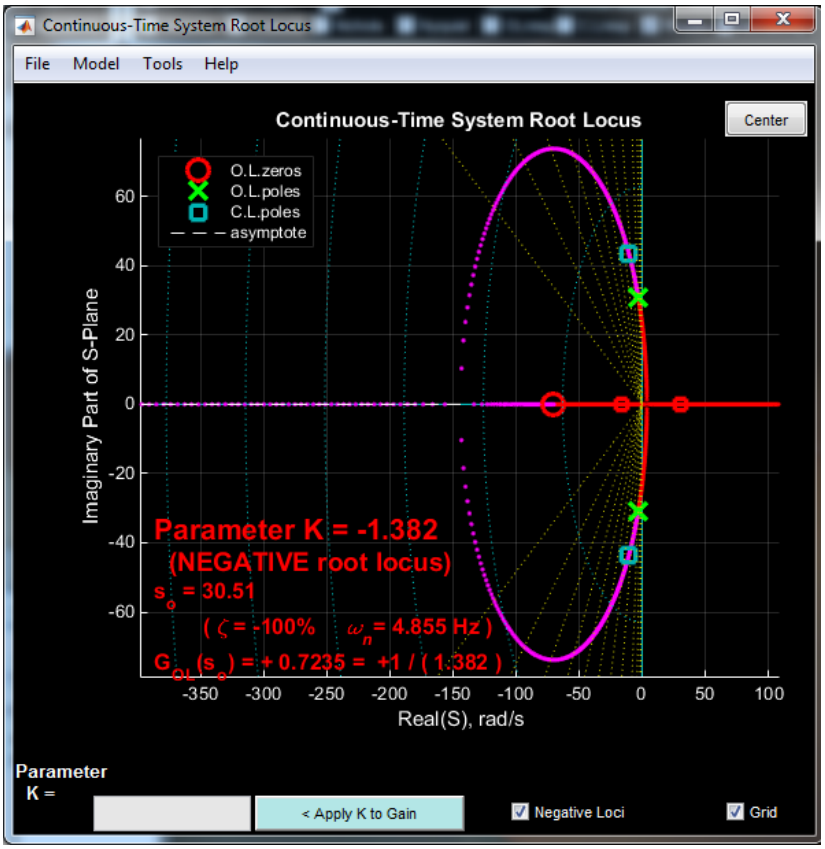


Alternatively, the gain "**5.215**" can be entered directly into the "**Parameter K =**" window.

If the "**< Apply K to Gain**" pushbutton is clicked, that factor will immediately be multiplied into the transfer-function gain that is currently in the main pole/zero interface.

The ***negative root-locus*** is an interesting aspect of the root-locus plot. Normally, the root-locus plot is generated only from the positive values of gain factor *K*. In **PZGui**, if the "**Negative Loci**" checkbox is selected, the complementary root-locus plot, based on negative values of *K*, becomes visible. The negative root-locus is plotted in red, to make it easily distinguishable from the ordinary (positive *K*) parts of the plot.

An example of this is shown in the following figure, in which the mouse-cursor has been brought close to one branch of the negative root-locus. The gain that would place a closed-loop pole at that point is negative, and is shown highlighted in red – again to emphasize that it is negative.



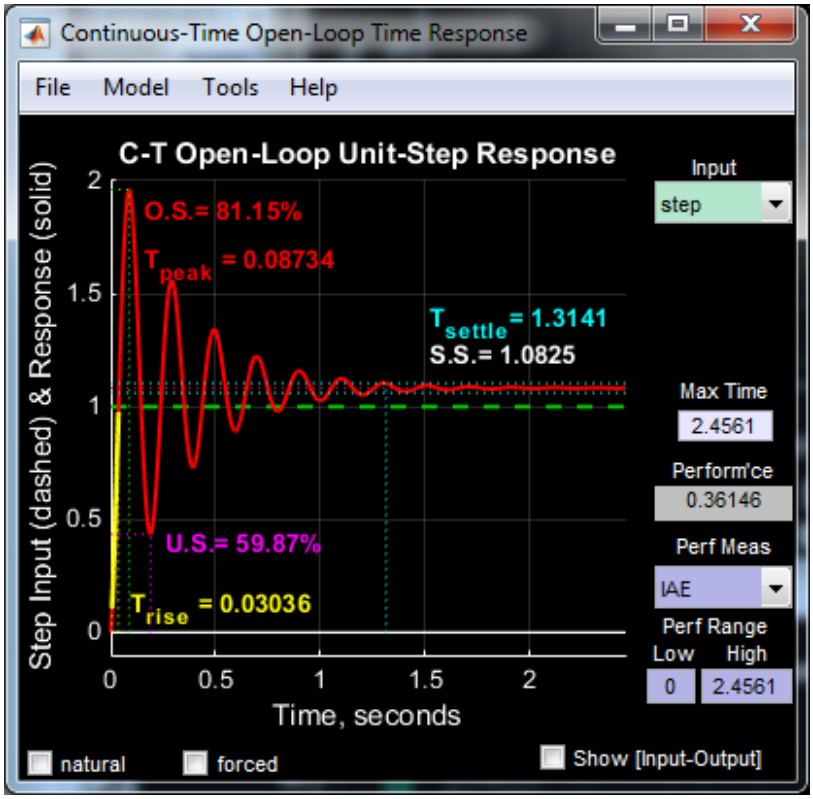
IX. *Time-Response Plots*: Open-Loop and Closed-Loop Response

On the **`PZGui`** main interface figure, there are two pushbuttons that create time-response plots.

(1) The open-loop time-response

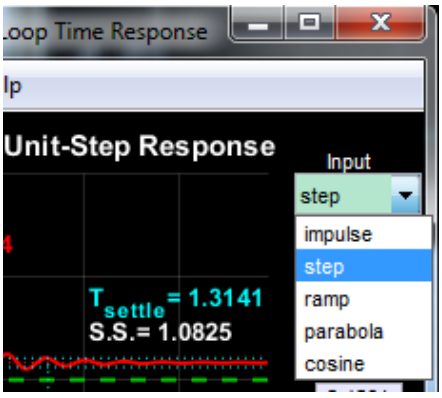
(2) The closed-loop time-response

Here is a typical time-response figure, in this case open-loop, with the (default) unit-step input.



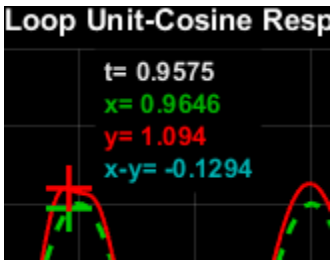
In each time-response plot, there is a dropdown menu to select any one of five different input signals:

- unit-impulse $\delta(t)$
- unit-step $u_s(t)$
- unit-ramp $tu_s(t)$
- unit-parabola $\frac{1}{2}t^2u_s(t)$, or
- unit cosine $\cos(\omega t)\,u_s(t)$, with adjustable frequency ω



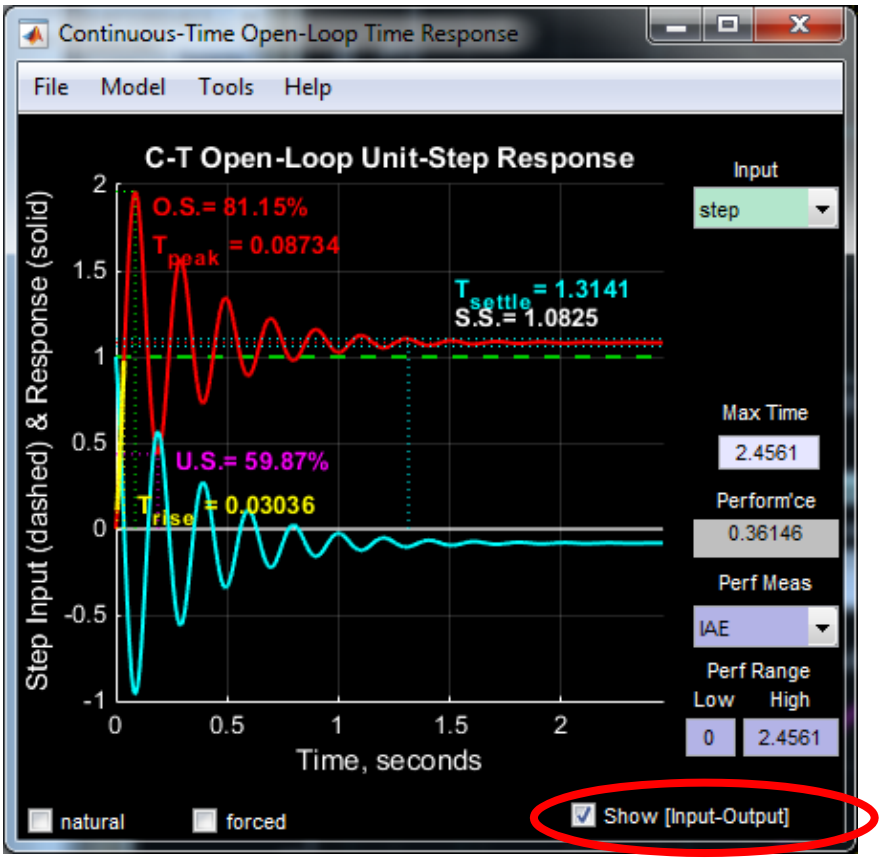
The maximum time of the time-response plot can be specified by simply editing the number that appears in the "**Max Time**" window. In this case, the final time is 2.4561 (seconds).

Whenever the mouse-cursor is inside the plot area, the values of input and output are highlighted. The time-axis location of the mouse is taken as the point of interest, and "crosshairs" appear at the corresponding time-points on the plot. In most of the plots, the difference, (input minus output) is also highlighted. Here is an example.



The Tracking Error

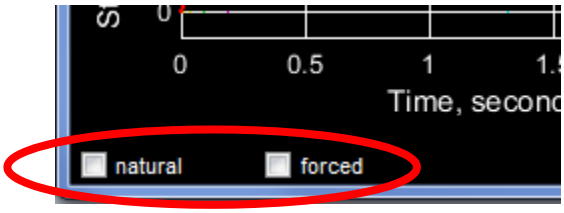
Sometimes in control design, one of the design goals is to achieve a certain maximum level of "tracking error," which is computed by subtracting the output from the input. The tracking error can be viewed by selecting the "**Show [Input-Output]**" checkbox, as shown here.



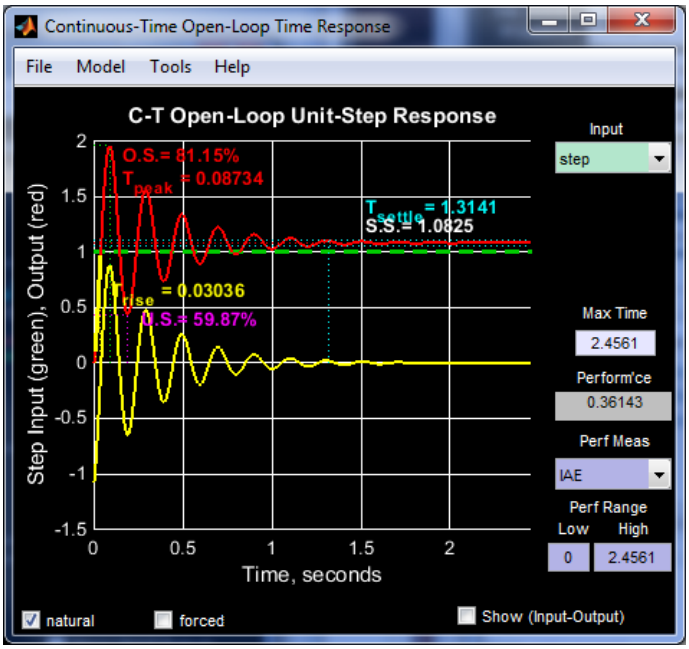
Display of tracking error is especially useful when the difference between them is very small.

The Natural Response and the Forced Response

All time-response figures have checkboxes, along the lower-left edge, that enable you to view the natural part of the response and the forced part of the response. When either or both of these are selected, the total response (*i.e.*, sum of natural-response plus forced-response) will still be visible.



Selecting the "**natural**" checkbox results in additional yellow lines in the plot, one for each decaying sinusoid (the natural response of each complex-conjugate pole pair in the system) and each decaying real exponential (the natural response of each real-valued pole in the system). For example, here is the plot for a second-order lightly-damped system.

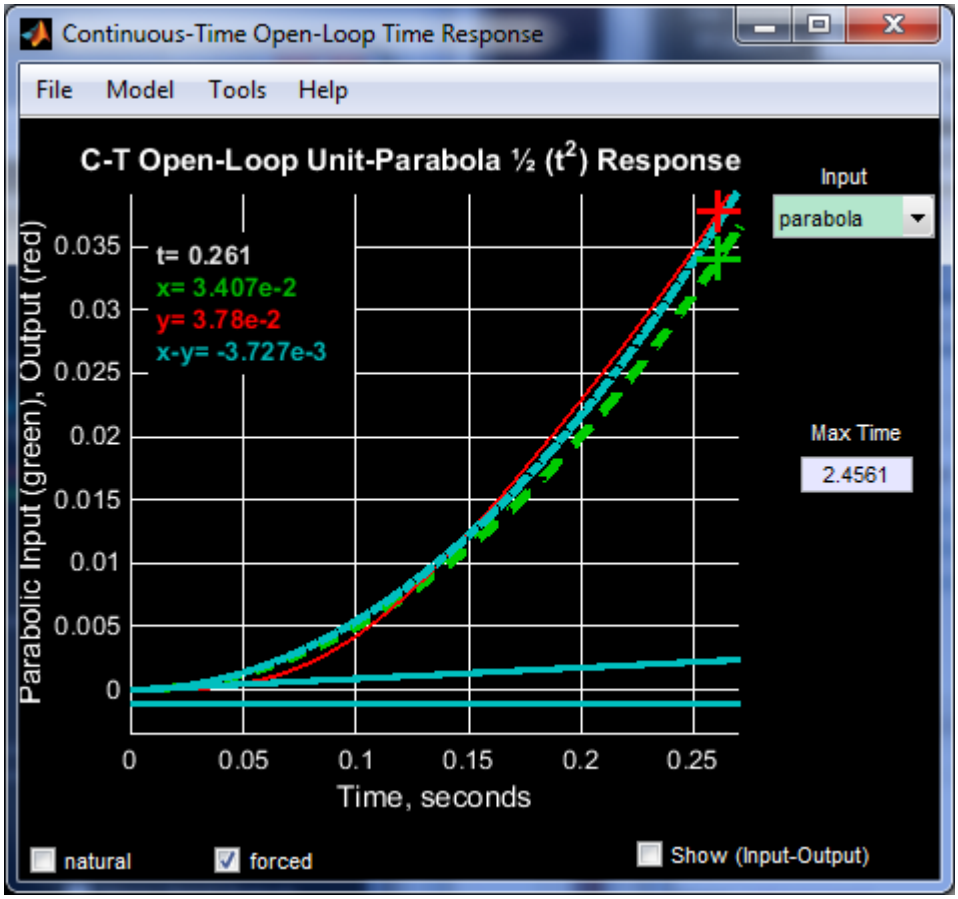


If there are more than 32 individual natural-response lines (*i.e.*, decaying exponentials and decaying sinusoids), they will not be plotted separately, because of the excessive graphics burden it can impose in the figure. Instead, the <u>total</u> natural response will be plotted as a single yellow line.

Selecting the "**forced**" checkbox results in an additional cyan-colored line that corresponds to the forced response. The forced-response is always "similar" to the input function, so for a unit-step input the forced response is also a step.

The forced-response is a little more complicated for unit-ramp input: because the input has two poles at dc, the forced response is actually a step-plus-a-ramp (two cyan lines). Similarly, the unit-parabola input has three poles at dc, so the forced response is a step-plus-a-ramp-plus-a-parabola (three cyan lines).

Directly below is an example of the open-loop response to a unit-parabola, showing the three cyan forced-response lines (because the "**forced**" checkbox is selected). The forced-response consists of a step at roughly -0.001, plus a ramp that increases to approximately 0.0025 during the quarter of a second covered by the plot, plus a parabola that ends up slightly below the red line of the output.



Some Input-Specific Features

(a) Unit-Impulse Input

Tracking error is not shown, because it is simply the negative of the output for all $t > 0$.

(b) Unit-Step Input

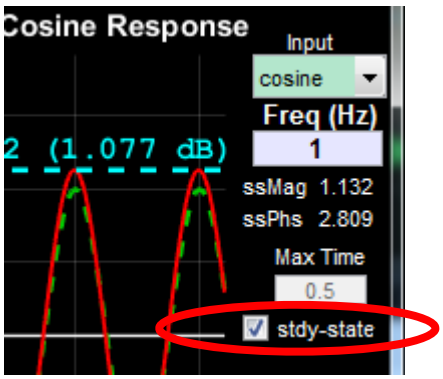
If the specified maximum time is at least (approximately) 20% longer than the settling time, then the step-response plot will automatically be annotated with rise time, peak time, percent overshoot, percent undershoot, settling time, and steady-state error.

A set of controls appears that will compute any one of four standard performance measures over any part of the step-response up to the specified maximum time. The selectable performance measures are: **IAE** (integrated absolute error), **ITAE** (integrated time by absolute error), **ISE** (integrated square error), and **ITSE** (integrated time by square error), selectable by a dropdown menu.

(c) Unit-Cosine Input

The default cosine frequency is 1 hertz, but this can easily be changed by way of the "**Freq (Hz)**" edit-window.

Steady-state magnitude and phase are displayed and, if the maximum time is at least (approximately) three times longer than the settling time, the plot is also annotated to graphically show these steady-state values. There is an option to show only the part of the plot around steady-state (specified maximum-time is disregarded).



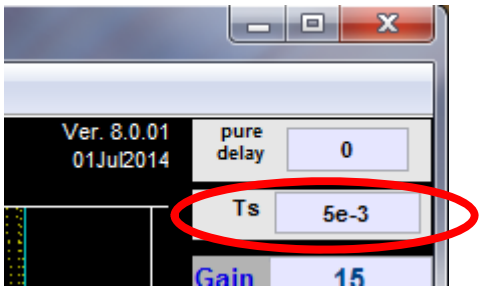
X. Linking the Continuous-Time and Discrete-Time Domains

In the study of control, it is often useful to view the "discrete-time equivalent" of a continuous-time system. For instance, a controller might be designed in continuous-time but implemented in discrete-time (*e.g.*, implemented by a microcontroller). **PZGui** has the capability to transform and "link" a continuous-time model into its "equivalent" discrete-time model.

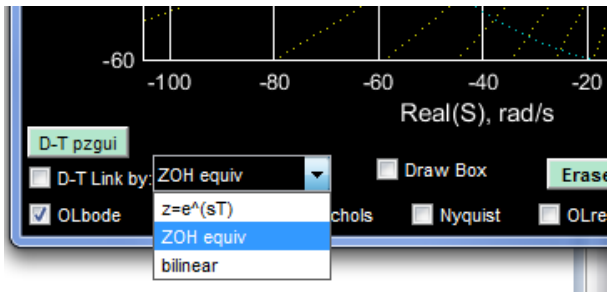
In **PZGui** there is a choice of three mathematical methods by which the transformation may be done.

- By simple mapping of the poles and zeros, according to the relation $z = e^{sT_s}$
- By zero-order-hold equivalence
- By the bilinear transformation

In any of these cases, the transformation is based on the sample period T_s that is specified in the main interface, as shown here:



The default is by zero-order-hold equivalence, but this can be changed using the dropdown menu, as shown here:

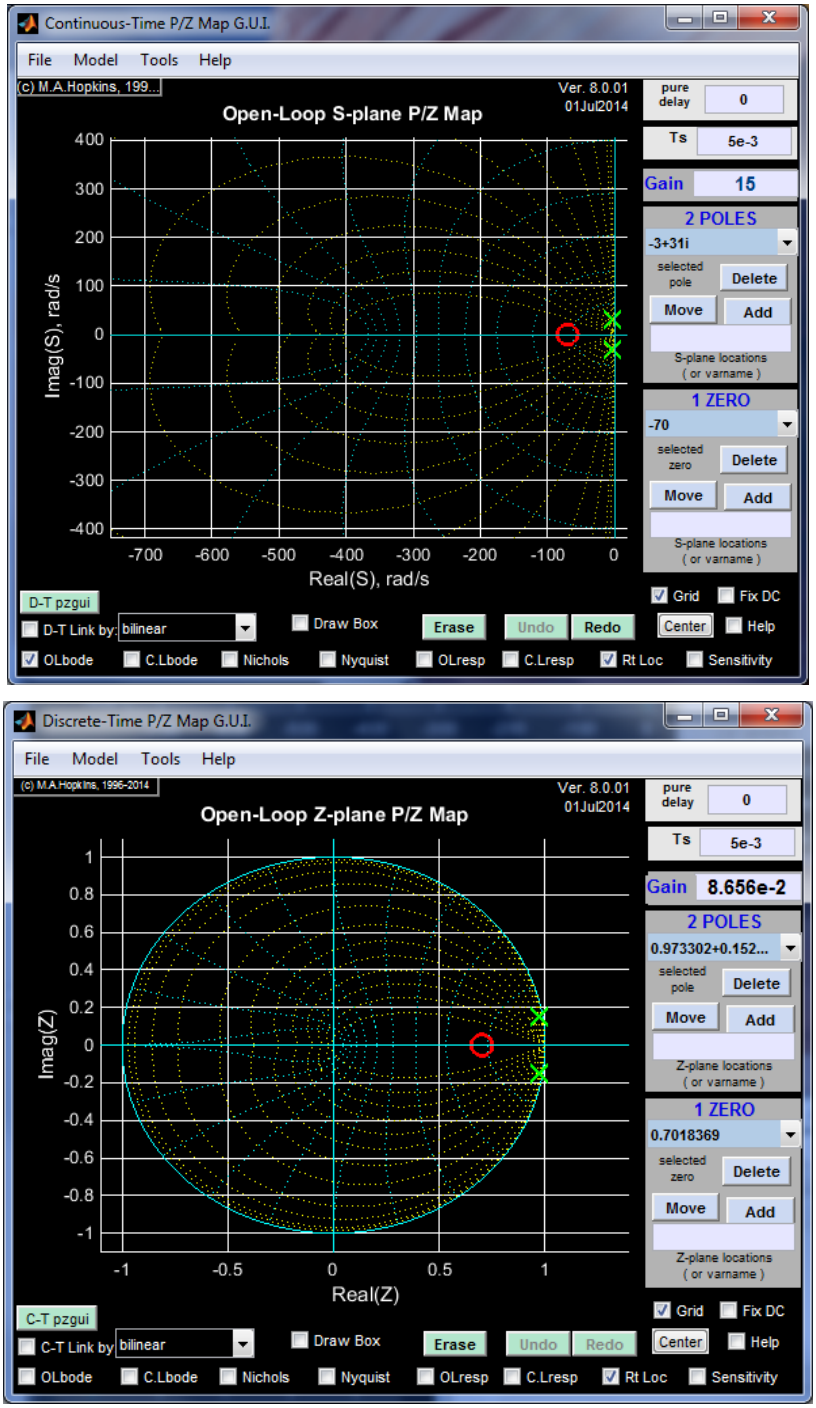


The link between continuous-time and discrete-time domains can be established in either direction, depending upon in which main interface figure the link is selected. For instance, the figure directly above is the continuous-time main interface, and checking the box labeled "**D-T Link by:**" will cause the continuous-time model to be mapped into the discrete-time domain, where it will replace any model that is currently in the discrete-time main interface.

If, instead, the corresponding link checkbox in the discrete-time main interface had been checked, then the ***inverse***-transformation of the discrete-time model into continuous-time would have been done, and the resulting continuous-time model would have replaced the model that was currently in the continuous-time interface.

Whenever a link is active (*i.e.*, one of the "link" checkboxes is selected), any change made to the transfer-function model that is being linked to the other domain will immediately result in a model having the corresponding change in the other domain.

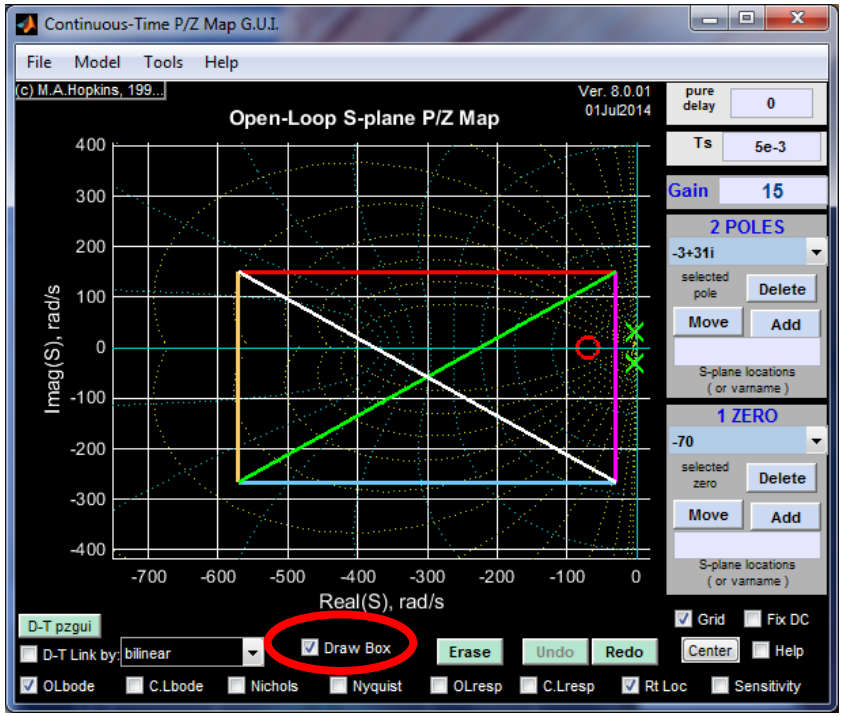
Even if no link is active, both of the main interfaces (*i.e.*, continuous-time and discrete-time) can be open at the same time. Whenever they are open at the same time, the continuous-time grid of constant damping lines and constant natural-frequency lines will correspond to the selected transformation method. Consider, for example, how the two main interfaces look when "**bilinear**" is selected.

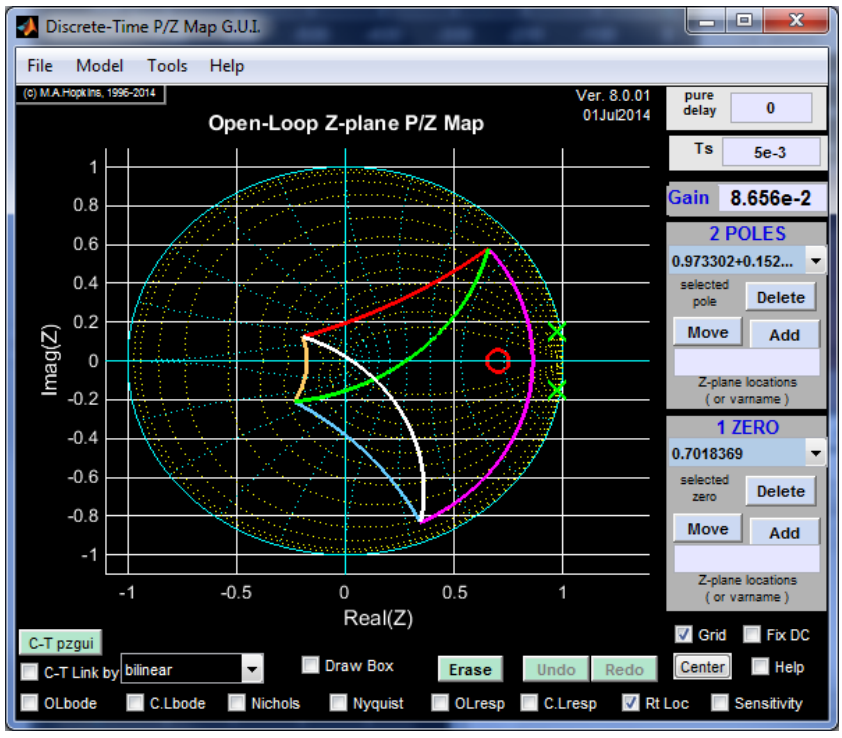


The continuous-time lines of constant damping factor are not straight, as they would otherwise be.

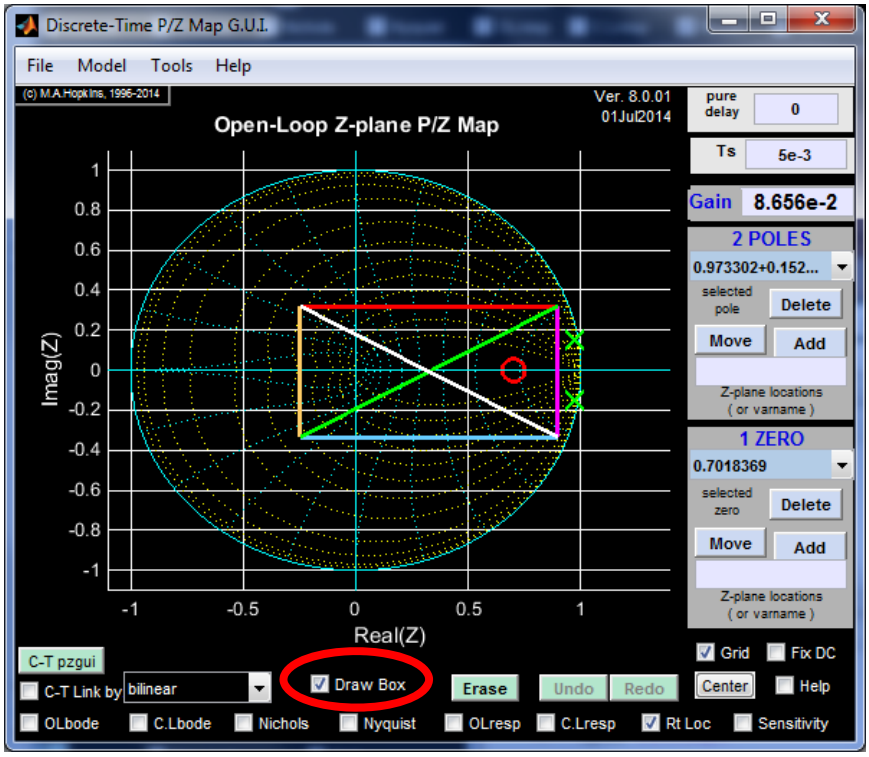
When both main interfaces are open, an interesting feature called "**Draw Box**" becomes available. When the "**Draw Box**" checkbox is selected, the mouse cursor becomes box-like, to remind you that you can't use the mouse to zoom, as you normally would. Instead, when you press-and-hold the left mouse-button, a box is drawn in the pole/zero map, and is also immediately mapped to the other domain by whichever link-method is selected. The boxes are also drawn simultaneously in the root-locus plots, if they are open.

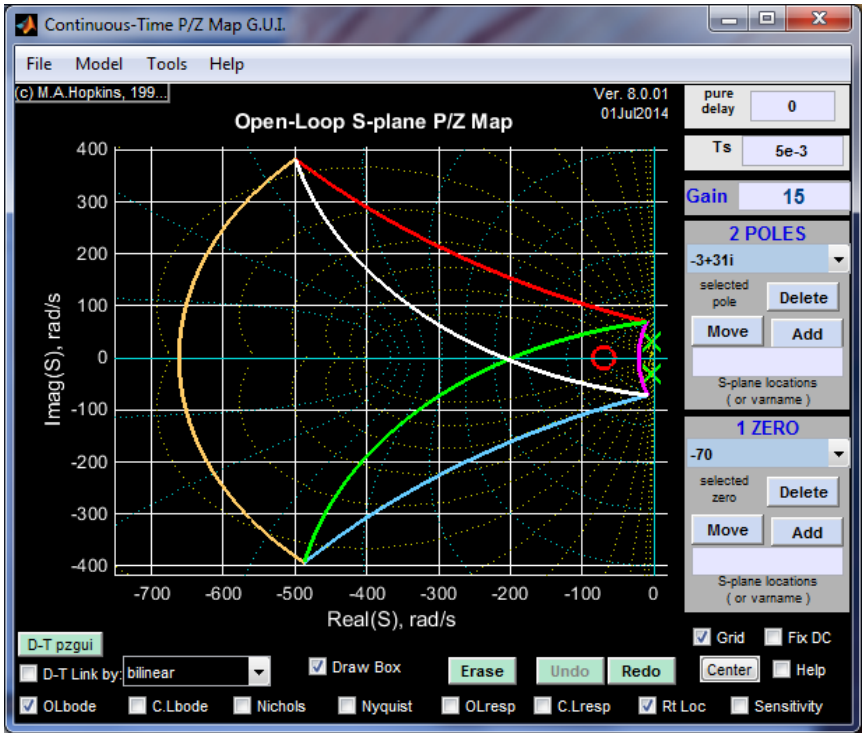
Here is a s-plane to z-plane example, with the bilinear transformation selected.



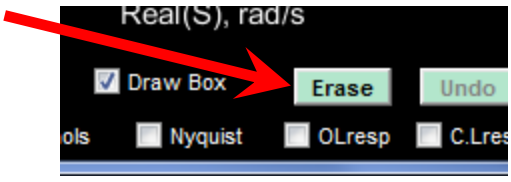


If the box is drawn in the discrete-time domain, instead, here is what results.



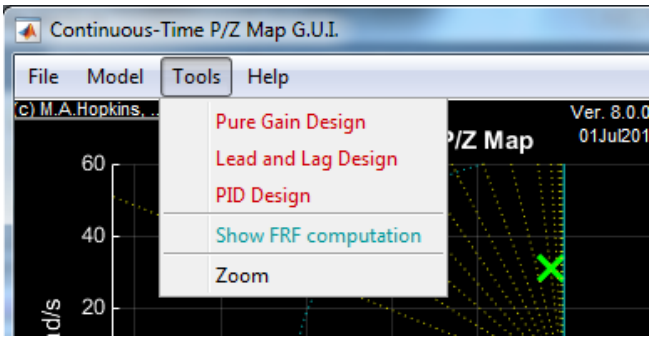


Any boxes you draw can be erased by the "**Erase**" pushbutton, in either main-interface figure.



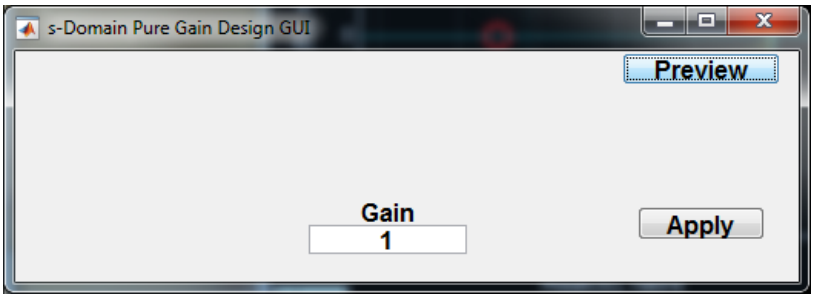
XI. The Tools Menu

The "**Tools**" menu is used to access the three major design tools in **PZGui**, as well as a tool that visually highlights the way that frequency-response is computed from the transfer function.

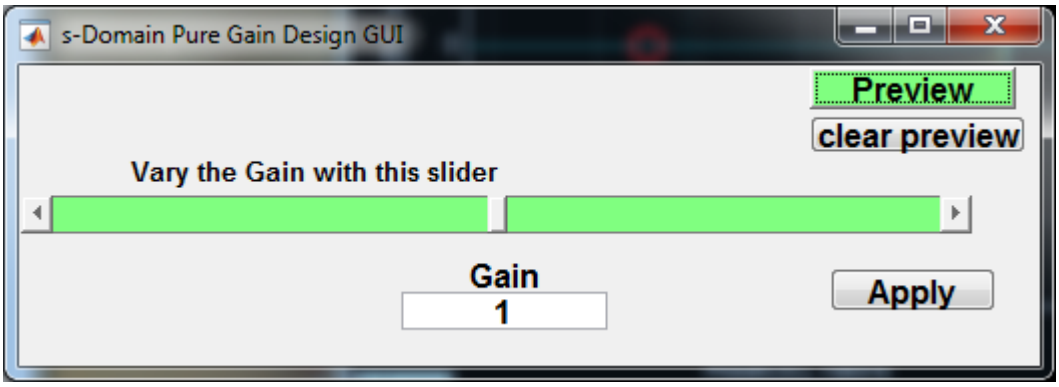


(1) The Pure Gain Controller Design Tool

The simplest possible feedback controller is a pure gain factor, $K > 0$. When teaching controls, for example when trying to teach root-locus methods, this is often the first type of controller to discuss. When the "**Pure Gain Design**" menu item is selected, the following interface figure will appear, with the default unity gain.

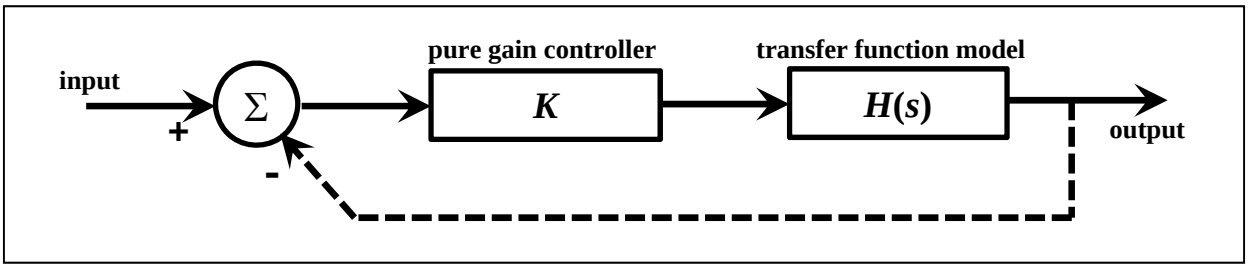


If any of the frequency-response or time-response plots are open and the "**Preview**" pushbutton is clicked, those plots will show previews of how they would look with the specified "Gain" in series with the zpk transfer function currently entered into **PZGui**.

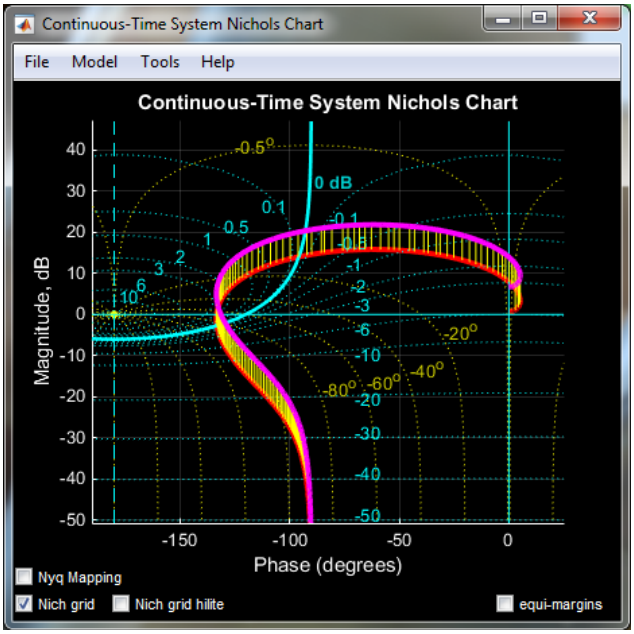


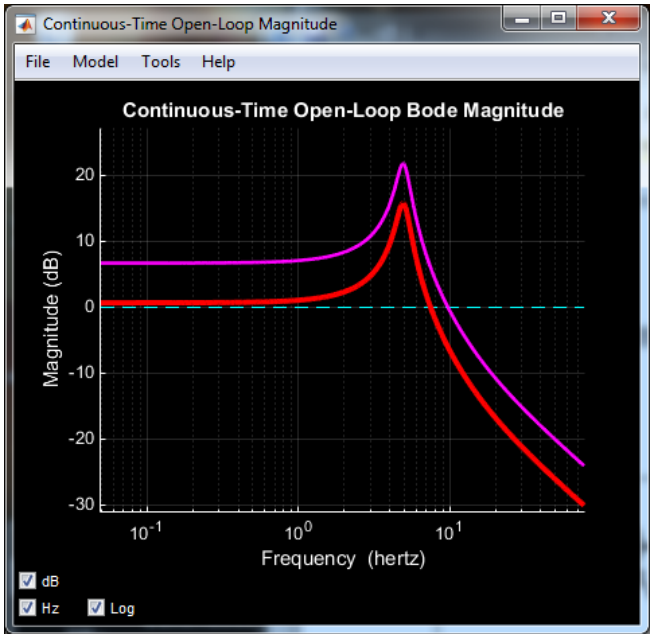
Gain can be changed by moving the slider, or by entering a different gain in the edit window.

For open-loop computations, the only effect is the change in dc gain, because pole and zero locations are unchanged. However, for closed-loop computations, the pure gain controller is assumed to be in the forward path, in series with the plant, as shown in the following block diagram. Thus, all closed-loop pole locations will change as the gain is varied.

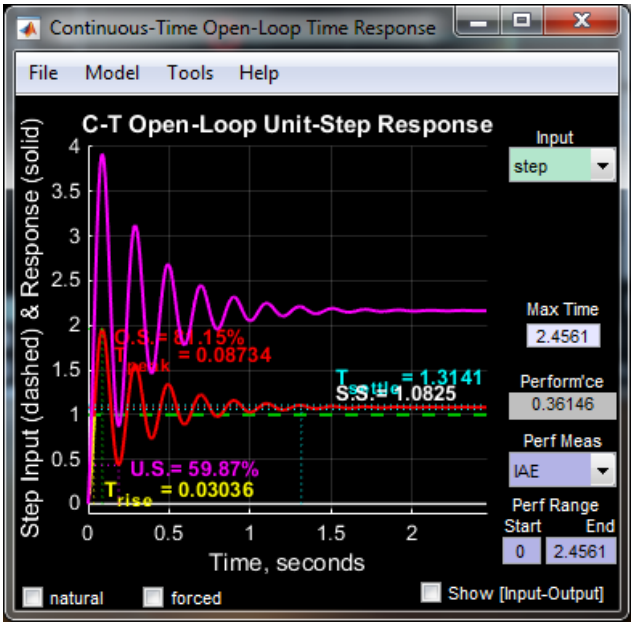


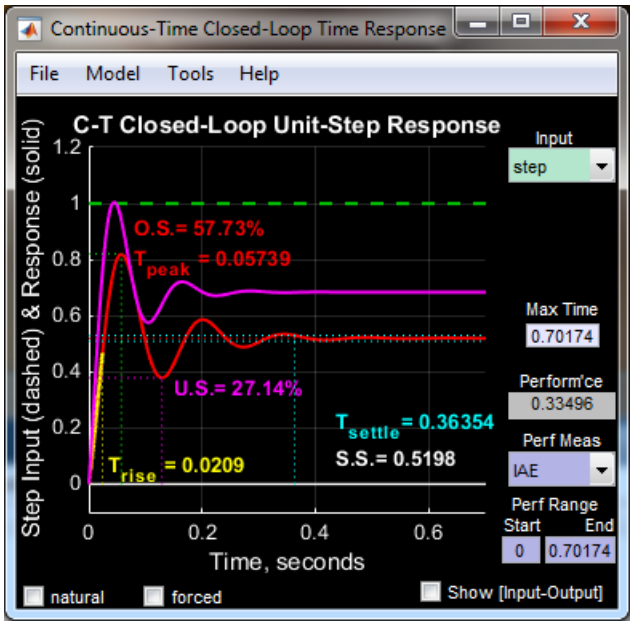
For example, here is the way two of the **PZGui** open-loop frequency-response plots look with "**Preview**" on and **Gain = 2**. **Preview lines are shown in magenta**. In the case of the Nichols chart, narrow yellow lines show how the individual frequency-points change (for pure gain-change, they simply move vertically).





For the same value, **Gain = 2**, here are the open-loop and closed-loop step-response previews.





Whether you change the **Gain** by entering a new value in the edit window, or by moving the slider, any open PZGui plots will "preview" the effect of that variation. If you use the slider, all previews will be updated in real time, while you are moving the slider.

If the "**clear preview**" pushbutton is clicked, preview mode is turned off, and the gain slider disappears.

If the "**Apply**" pushbutton is clicked, the value of gain currently in the design tool will be multiplied into the transfer function gain (*i.e.*, "applied") in the main pole/zero interface, and preview mode will be turned off.

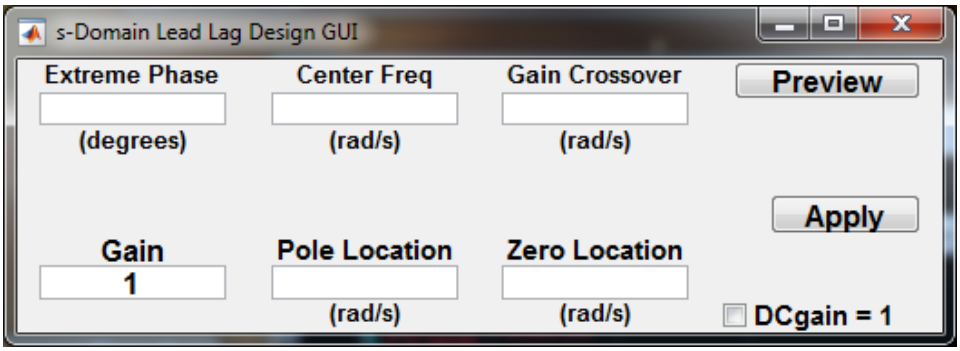
(2) The Lead-Controller and Lag-Controller Design Tool

Both the lead-controller and the lag-controller consist of one pole, one zero, and a gain factor,

$$G(s) = \frac{K(s-\xi)}{s-p}$$

where the pole p and zero ξ must be non-positive real numbers (*i.e.*, not greater than zero), and $K > 0$. If pole location $p > \xi$, the controller is a phase-lag, or lag controller. If pole location $p < \xi$, the controller is a phase-lead, or lead controller.

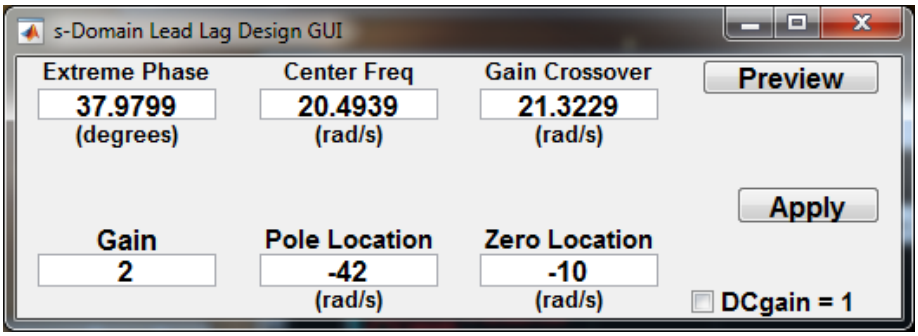
When the "**Lead and Lag Design**" menu item is selected, the following interface figure will appear, with the default unity gain, and no zero or pole specified.



The controller may be specified either

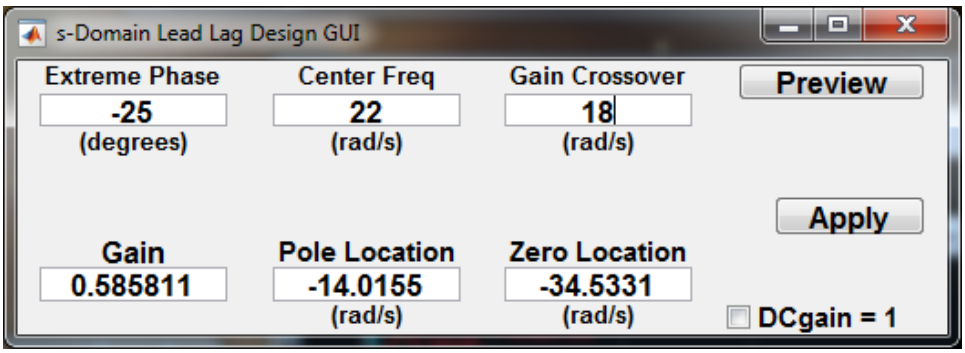
- in terms of its pole and zero, or
- in terms of its phase characteristics: extreme phase value, and frequency at which that occurs

For example, specifying a controller ***p*** = **-42**, ξ = **-10**, and **Gain** = **2** results in the following figure.



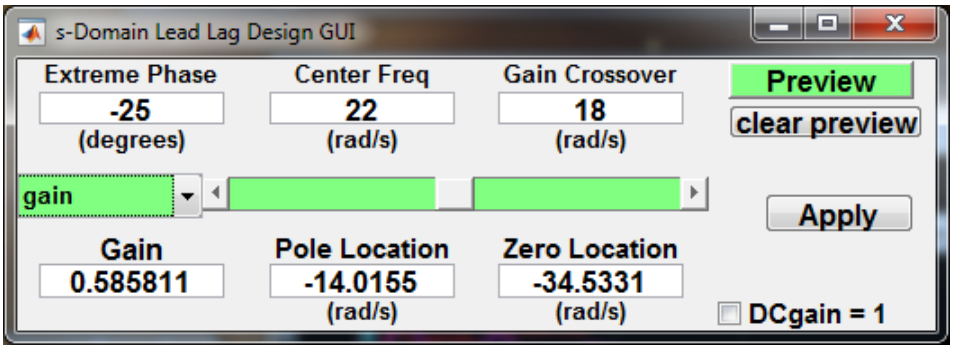
Notice that the **upper** row of parameters has been calculated and entered automatically.

As another example, specifying a controller that has an extreme phase of -25°, a center frequency of **22** radians/second, and a gain-crossover frequency of **18** radians/second results in the following figure.

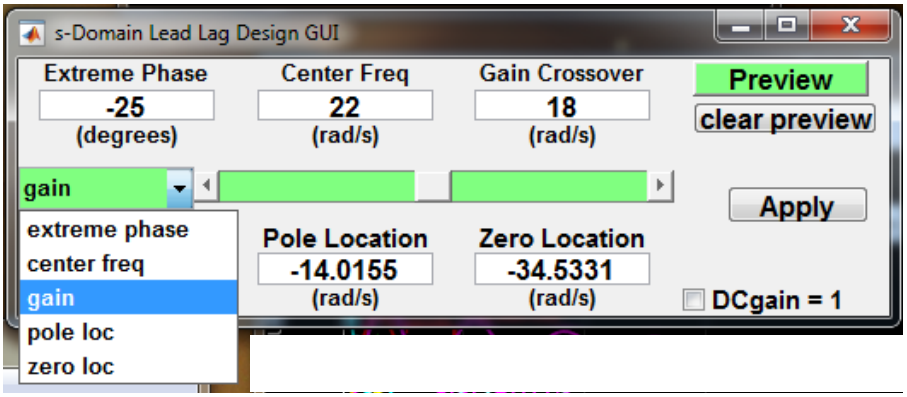


Notice that the **lower** row of parameters has been calculated and entered automatically.

If the "**Preview**" pushbutton is clicked, it puts the design tool in ***preview mode***, and it now appears as in this figure.



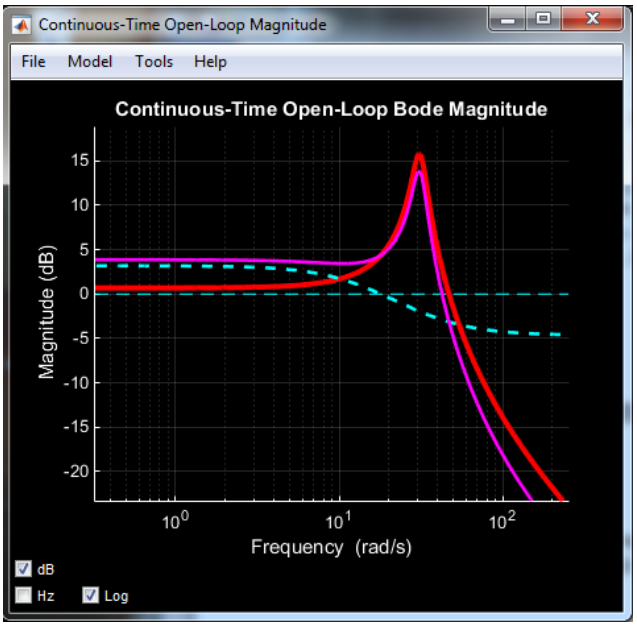
The green-colored slider in the center enables you to smoothly vary whatever parameter is selected in the dropdown menu to its left. All parameters but the gain-crossover frequency can be varied this way.

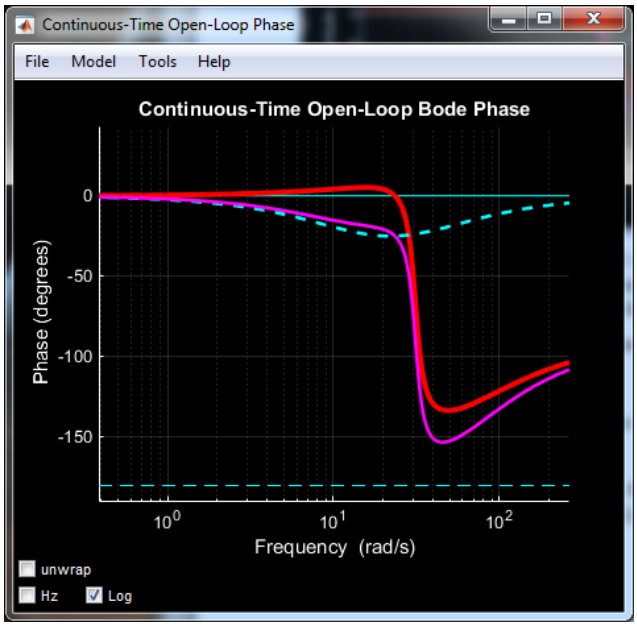


Whether you change a parameter by entering a new value in the appropriate edit window, or by selecting that parameter in the menu and using the slider, any open **`PZGui`** plots will "preview" the effect of that parameter value in the controller. If you use the slider, all previews will be updated in real time, while you are moving the slider.

The preview lines are generally magenta, but in the open-loop Bode plots, there is ***additional*** information about the controller design. In particular, the open-loop Bode magnitude plot shows the magnitude plot of the controller, itself, as a dashed cyan line. Likewise, the open-loop Bode phase plot shows the phase plot of the controller, itself, as a dashed cyan line.

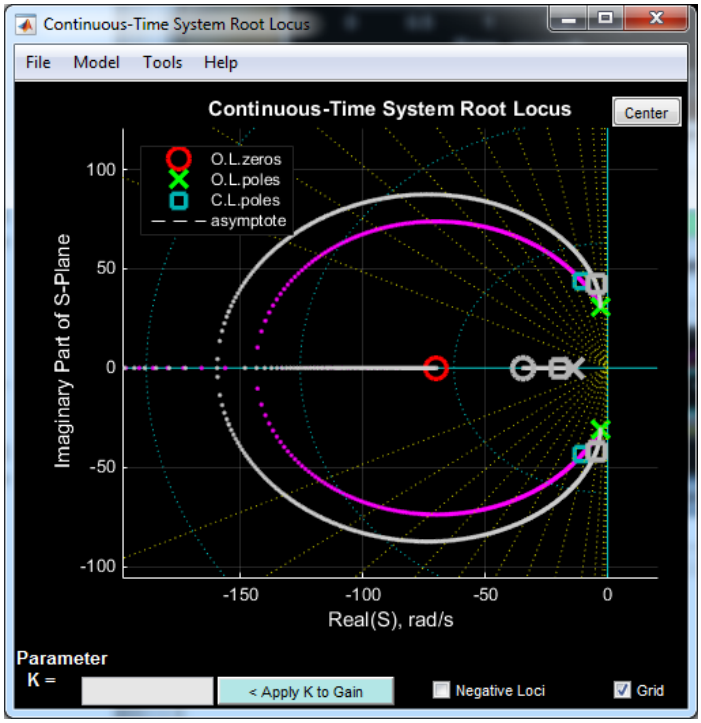
As an example, here are the open-loop Bode plots previewing a phase-lag design with extreme phase of -25°, a center frequency of **22** rad/s, and a gain-crossover frequency of **18** rad/s.



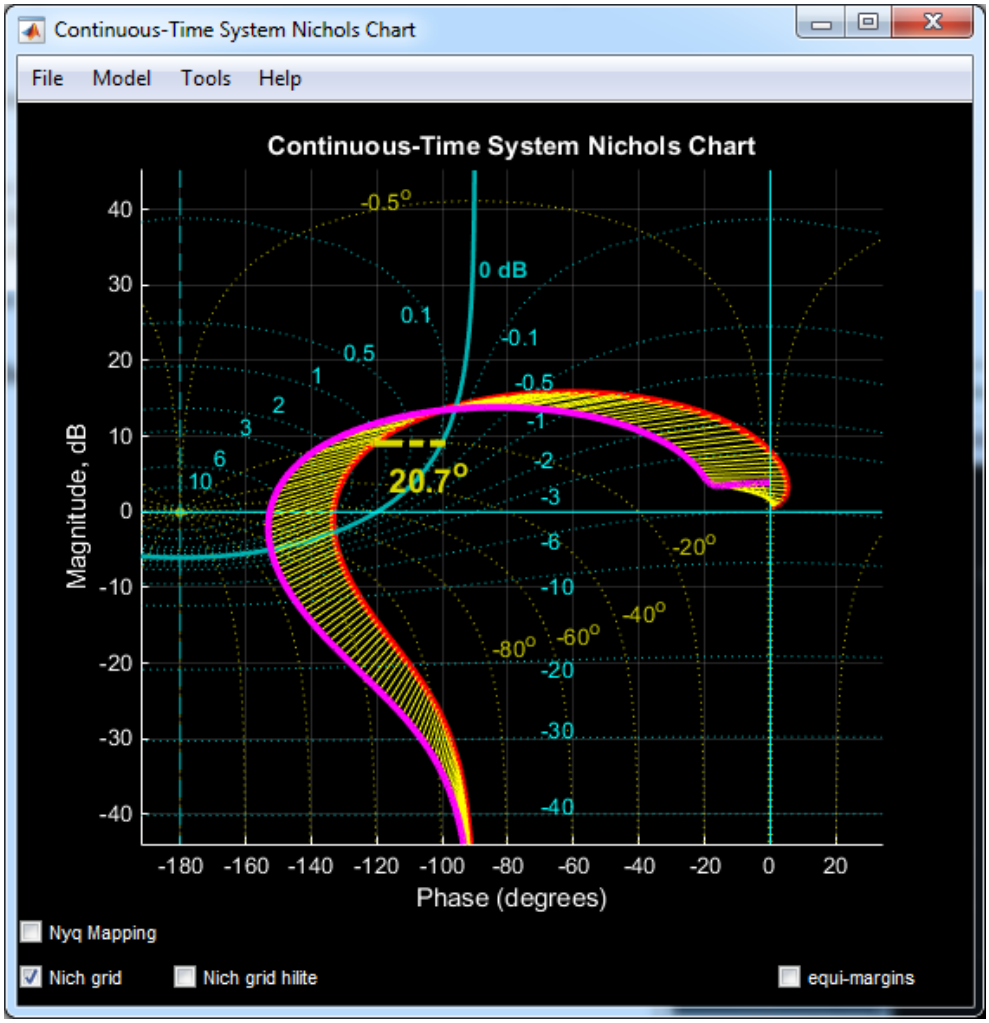


The red lines are the uncompensated system (the only lines in the plot when preview mode is turned off). The magenta lines are the preview lines of the compensated system. The dashed cyan lines are the magnitude and phase of the phase-lag compensator, by itself.

For that same phase-lag controller, here is the root-locus plot, with its preview. Note that, because the basic plot is magenta-colored, the preview plot color is gray, including the additional pole and zero of the controller



Again for that same phase-lag controller, here is the Nichols chart, with its preview. Notice the thin yellow lines that show the way individual frequency-points are changed by this controller.



If the "**clear preview**" pushbutton is clicked, preview mode is turned off, and the gain slider disappears.

If the "**Apply**" pushbutton is clicked, the values of pole, zero, and gain currently in the design tool will be worked into the open-loop transfer function (*i.e.*, "applied") that is currently in the main pole/zero interface, and preview mode will be exited.

As usual, you can undo the "apply" step using the "**Undo**" pushbutton in the main pole/zero interface.

(3) The PID Controller Design Tool

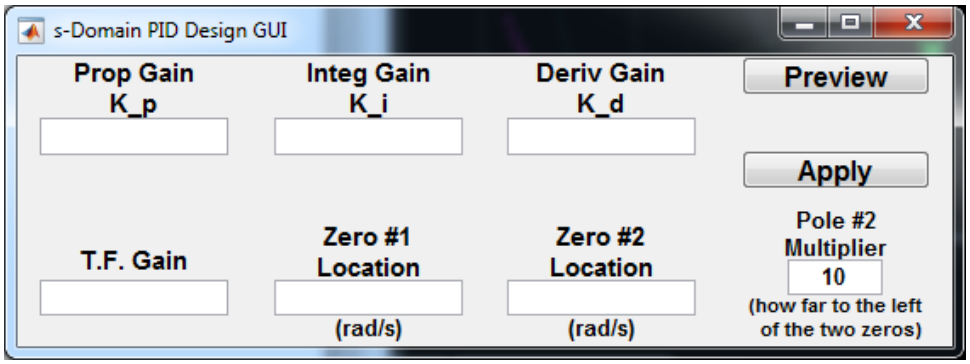
The PID-controller is specified by the following transfer function,

$$G(s) = \frac{K(s-\xi_1)\,(s-\xi_2)}{s\,(\frac{s}{p_2}-1)}$$

where the pole p_2 and zeros ξ_1 and ξ_2 must be in the left-half of the s-plane (for discrete-time, inside the unit-circle), and $K > 0$. The two zeros may be real-valued, or they may be a complex-conjugate pair.

The second pole p_2 is included in the PID transfer function for only one reason**:** so that the PID transfer-function will be ***proper***. In the *s*-plane, p_2 is typically much farther to the left than either of the zeros. In the *z*-plane, it is placed at $z = 0$. The transfer function denominator factor $(\frac{s}{p_2}-1)$ is expressed in that form in order to decouple the dc gain computation from changes in the location of p_2.

When the "**PID Design**" menu item is selected, the following interface figure will appear, with no gains, and no zeros specified.

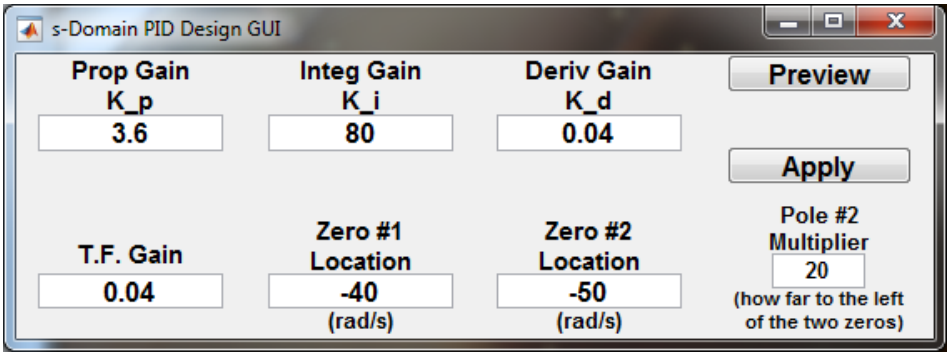


Notice that the location of p_2 is always relative to the locations of the zeros. This pole location is specified as a multiple of minus the absolute value of the left-most zero location. The default multiple is 10, but it can be specified anywhere from 3 to 1000.

The controller may be specified either

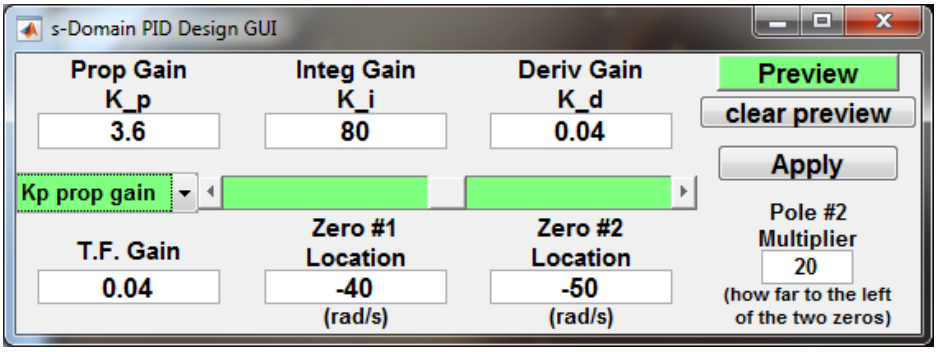
- in terms of its two zeros, second pole, and transfer-function gain, or
- in terms of its PID gains: proportional gain K_p, integral gain K_i, and derivative gain K_d

For example, specifying a controller ξ_1 = **-40**, ξ_2 = **-50**, **T.F. Gain** = **0.04**, and **Pole #2 Multiplier** = **20**, results in the following figure. That puts p_2 at *s* = **-50 x 20 = -1000.**

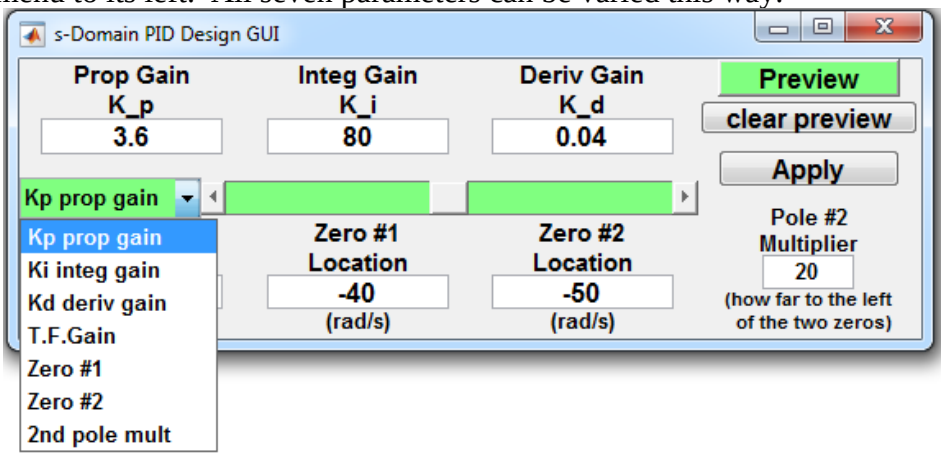


Notice that the **upper** row of gain parameters has been calculated and entered automatically, based on those entered in the lower row.

If the "**Preview**" pushbutton is clicked, it puts the design tool in ***preview mode***, and it now appears as in this figure.

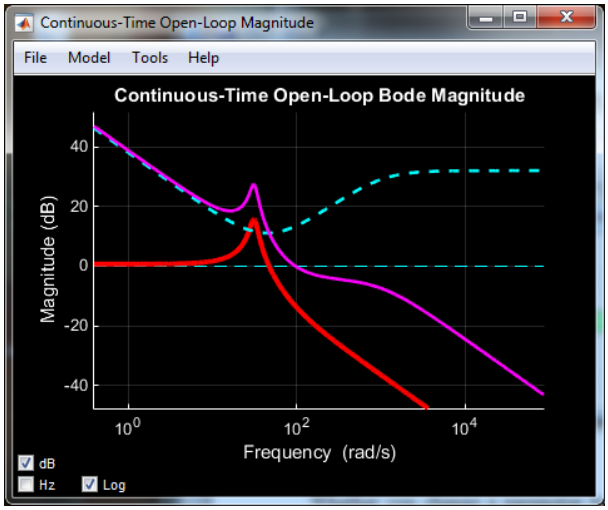


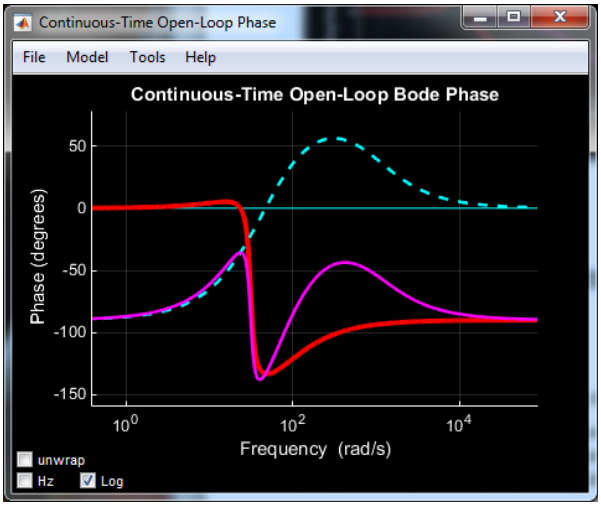
The green-colored slider in the center enables you to smoothly vary whatever parameter is selected in the dropdown menu to its left. All seven parameters can be varied this way.



Whether you change a parameter by entering a new value in the appropriate edit window, or by selecting that parameter in the menu and using the slider, any open **`PZGui`** plots will "preview" the effect of that parameter value in the controller. If you use the slider, all previews will be updated in real time, while you are moving the slider.

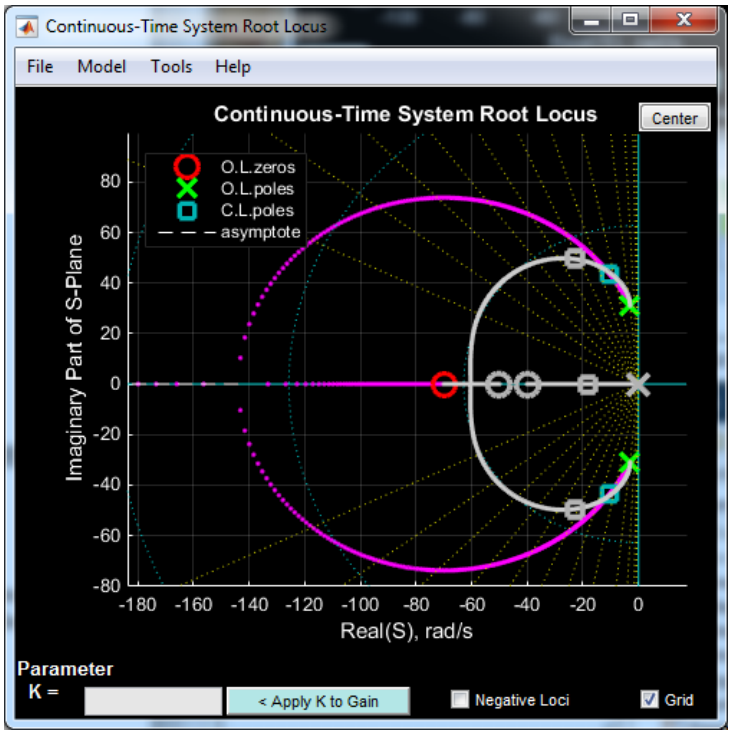
The preview lines are generally magenta, but in the open-loop Bode plots, there is ***additional*** information about the controller design. In particular, the open-loop Bode magnitude plot shows the magnitude plot of the controller, itself, as a dashed cyan line. Likewise, the open-loop Bode phase plot shows the phase plot of the controller, itself, as a dashed cyan line.



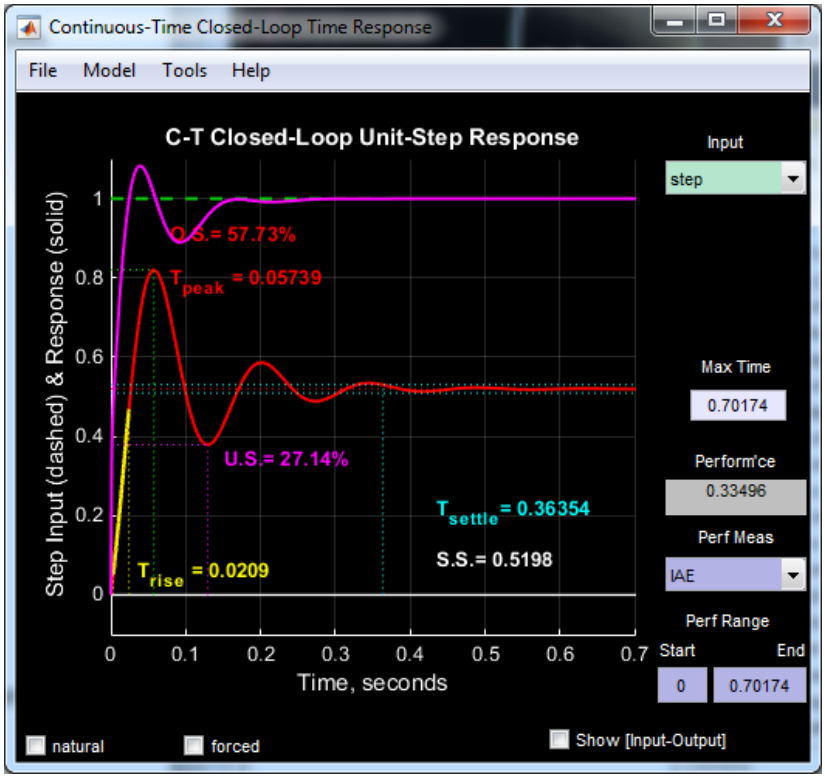


The red lines are the uncompensated system (the only lines in the plot when preview mode is turned off). The magenta lines are the preview lines of the compensated system (*i.e.*, with the controller). The dashed cyan lines are the magnitude and phase of the phase-lag controller, by itself.

Here is the corresponding root-locus preview. The preview root locus is in gray, and so are the poles and zeros of the PID controller.

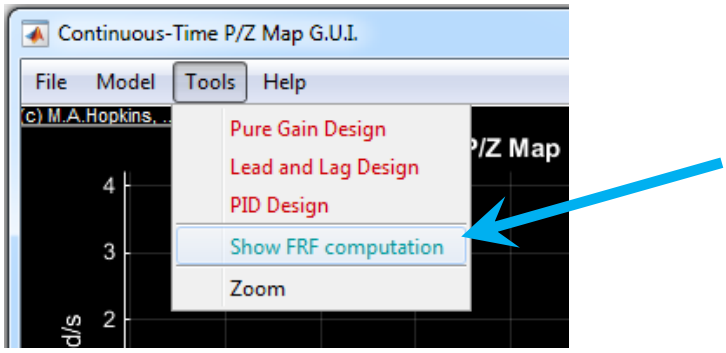


Here is the corresponding closed-loop step-response preview (the magenta line). Note that the annotations still refer to the uncompensated response (the red line).



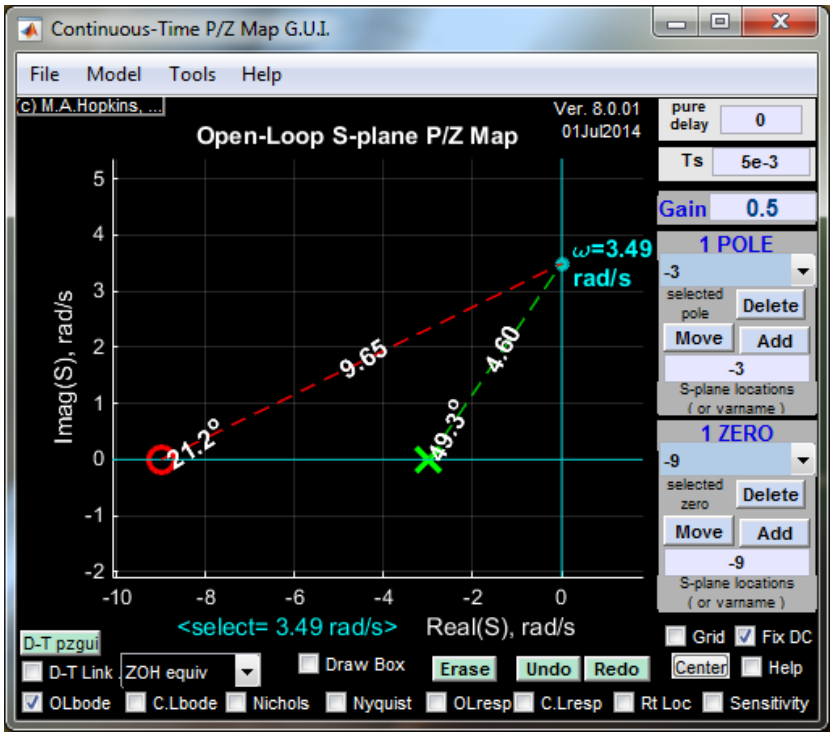
(4) Illustrating the Frequency-Response Computation

This tool can help students understand how computation of the unit-cosine response (frequency-response) at any frequency ω is related to vectors drawn from the various poles and zeros of the system to the point on the positive imaginary axis, $s = j\omega$.



Here is what happens when the "**Show FRF Computation**" menu item is selected, in the "**Tools**" menu. If the cursor is placed near the positive imaginary-axis in the main pole/zero interface, the nearest point on the axis is highlighted by a small cyan circle. Furthermore, vectors are drawn to that point from the various poles and zeros in the pole/zero map, and those vectors are ***annotated*** with their angles and their lengths. Those angles and lengths can be used to compute the frequency-response magnitude and angle at the highlighted frequency.

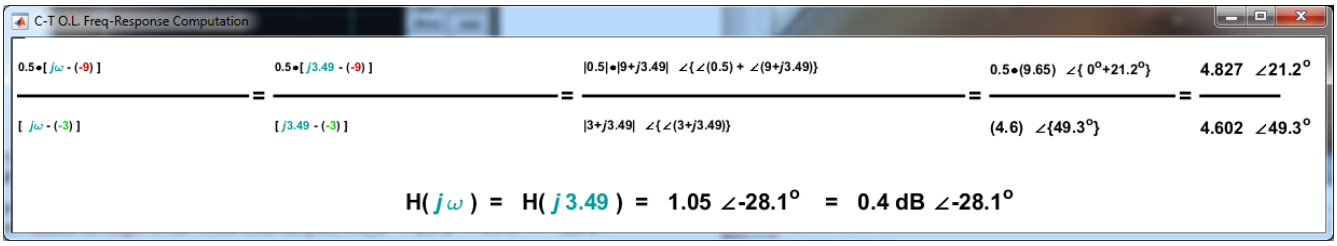
For example, with a pole at **-3** and zero at **-9**, the cursor was placed near to s = **3.49***i*. So ω = 3.49 radians/second. The following figure shows the resulting vectors and highlights.



The frequency-response magnitude at **3.49** rad/s is the gain **times** the length of the vector from the zero and **divided by** the length of the vector from the pole, i.e., **0.5** x **9.65** / **4.60** = **1.05**.

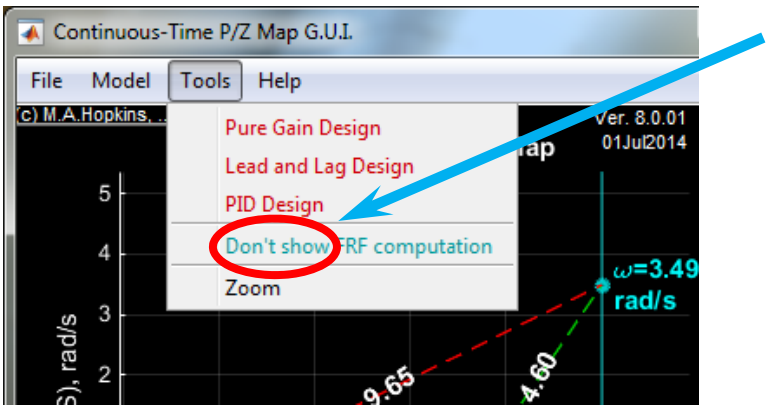
Also, the frequency-response phase at that frequency is the angle of the gain **plus** the angle of the vector from the zero **minus** the angle of the vector from the pole, *i.e.*, $\mathbf{0^\circ + 21.2^\circ - 49.3^\circ = -28.1^\circ}$.

If the system doesn't exceed third-order, the ***details*** of the computation are also shown, in a separate figure. In this case, because the system is only first-order, the following separate figure appears.

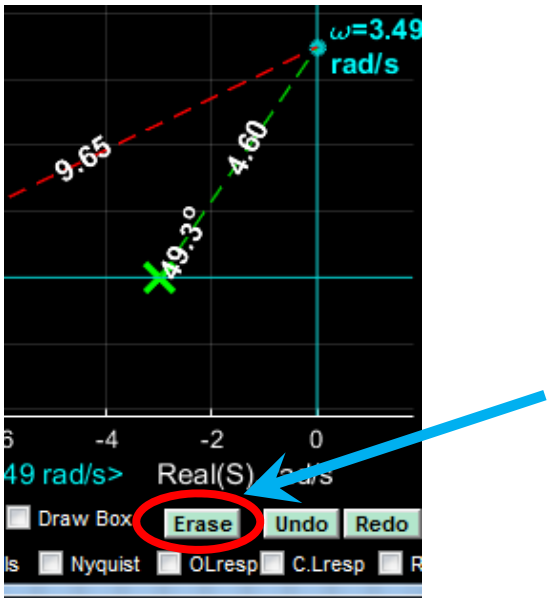


There are two ways to **close this tool** and get rid of the highlights, vectors, and computation-figure.

First, when this tool is open the "**Tools**" menu looks slightly different: now one of the items you can select is "**Don't show FRF computation**". Note that the same "**Tools**" menu is available in any of the **`PZGui`** plots, as well as the main interface figure.



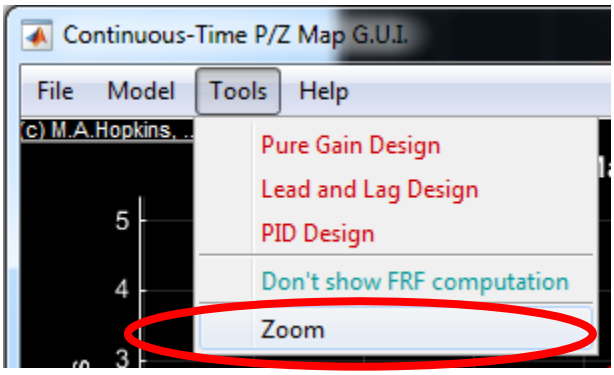
Second, you can also turn off this tool using the "**Erase**" pushbutton on the main pole/zero interface.



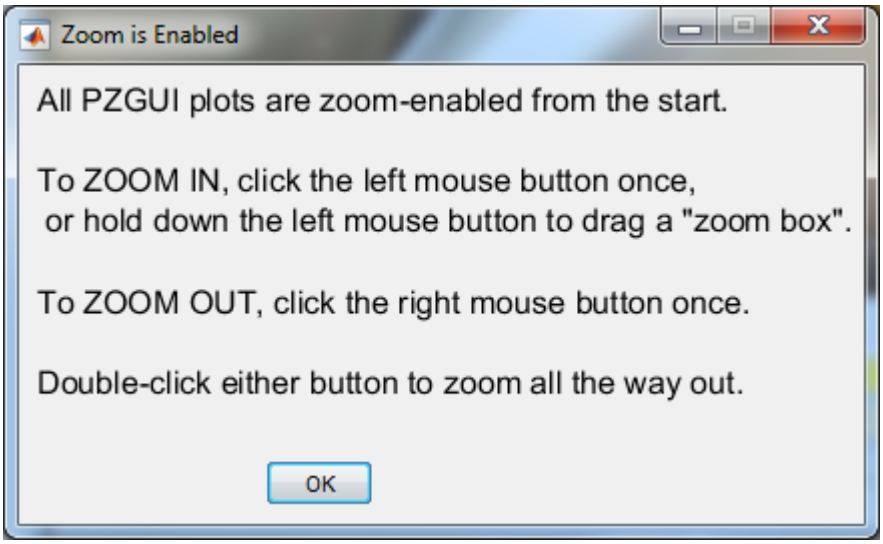
This tool works similarly in the discrete-time domain, except that it does not produce a figure that shows the details of the *z*-plane computation. That figure is only generated for the *s*-plane computation.

(5) Zooming In and Out in the Plots

All plots, by default, are completely zoom-enabled. If you click on the "Zoom" menu item,



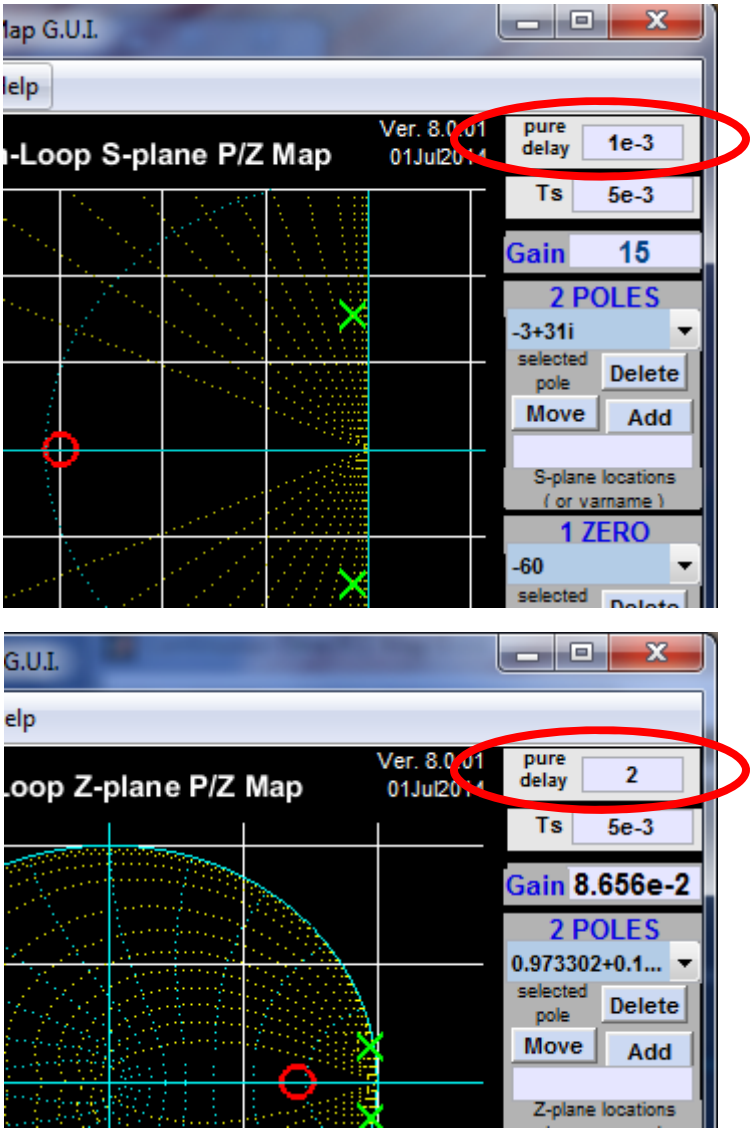
the following advisory will appear, explaining how to use the mouse to zoom in and out, in the plots.



XII. Adding "Pure Delay" to a Model

The **PZGui** tool has the capability to model ***pure delay*** in systems. For continuous-time systems, the amount of pure delay is specified in seconds. For discrete-time systems, delay is specified in terms of the integer number-of-samples of delay.

The default value of delay is zero, but this is easily changed by entering a different value into the edit window at the upper-right corner of the main pole/zero interface figure, as shown below. Notice that discrete-time (*Z*-plane) pure delay must be an integer. Of course, delay must be non-negative.



For discrete-time models it is straightforward to model the effects of an integer number-of-samples of delay, because each sample of delay corresponds to the addition of another pole at $z = 0$.

However, for continuous-time models, it is not as simple. Difficulties arise in two cases: the closed-loop time-response plot (simulation), and the root-locus plot. In these two cases, delay is handled by introducing the standard fourth-order Pade-approximation of delay.

The algorithm used by Matlab® for Pade approximation of delay is from Golub and VanLoan, Matrix Methods, 3rd ed., pp. 572-574, and for a fourth-order approximation it is

$$e^{-T_o s} \cong \frac{s^4 - \frac{20}{T_o}s^3 + \frac{180}{T_o^2}s^2 - \frac{840}{T_o^3}s + \frac{1680}{T_o^4}}{s^4 + \frac{20}{T_o}s^3 + \frac{180}{T_o^2}s^2 + \frac{840}{T_o^3}s + \frac{1680}{T_o^4}}$$

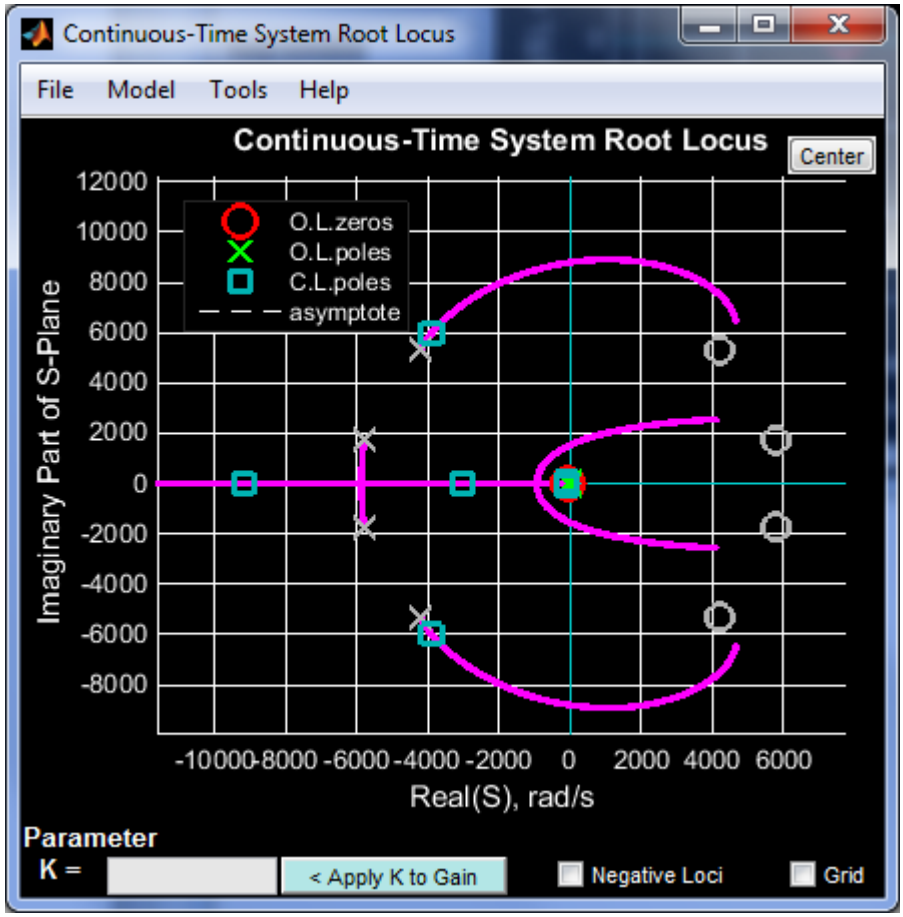
In effect, this introduces four more poles and four more zeros into the transfer function.

This approximation is only used for the closed-loop time-response, and the root-locus plot.

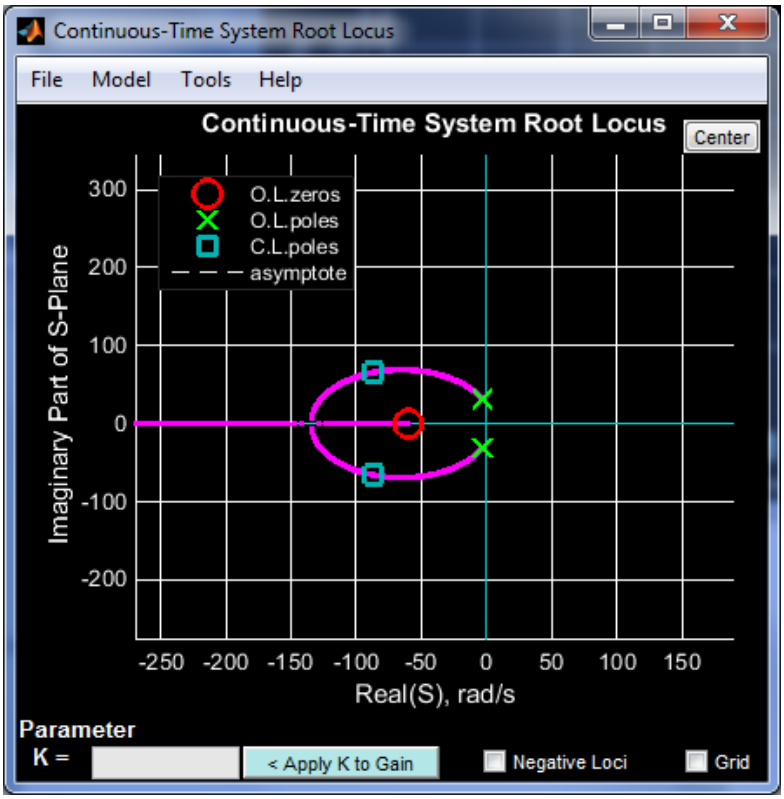
While use of the Pade approximation in the closed-loop time-response plot is not obvious, in the root locus plot it is *quite* obvious, because four additional poles and four additional zeros appear in the plot. These Pade poles and zeros are colored gray in the plot, to emphasize that they are not ordinary poles and zeros.

For relatively small amounts of delay, the Pade poles and zeros tend to be far from the dominant poles and zeros of the transfer function, as shown in the following examples. The first example has only a small amount of delay, but the second example has much more significant delay.

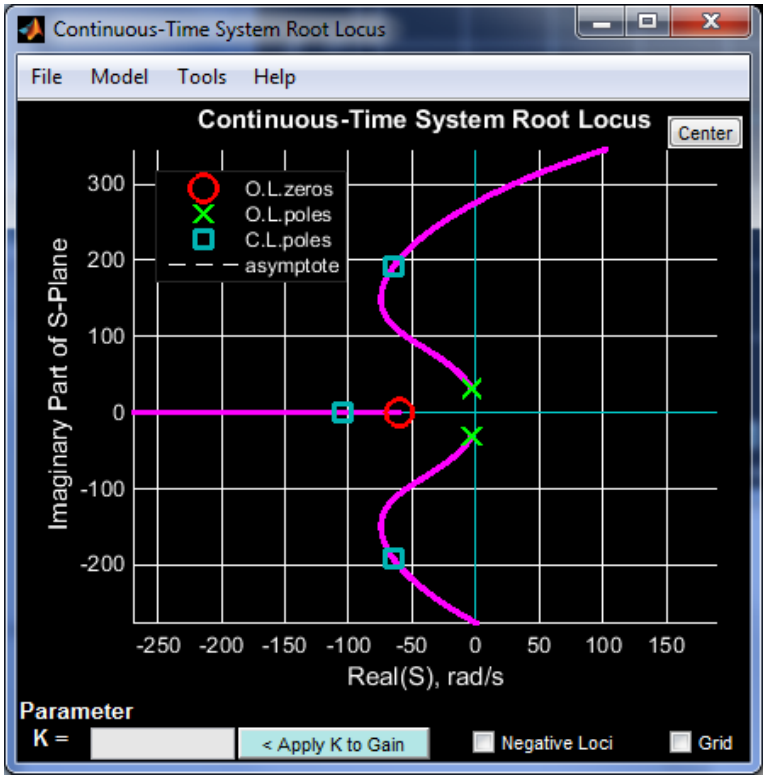
The first plot is "zoomed out" to the point where the Pade poles and zeros are clearly visible. The delay is set to one millisecond.



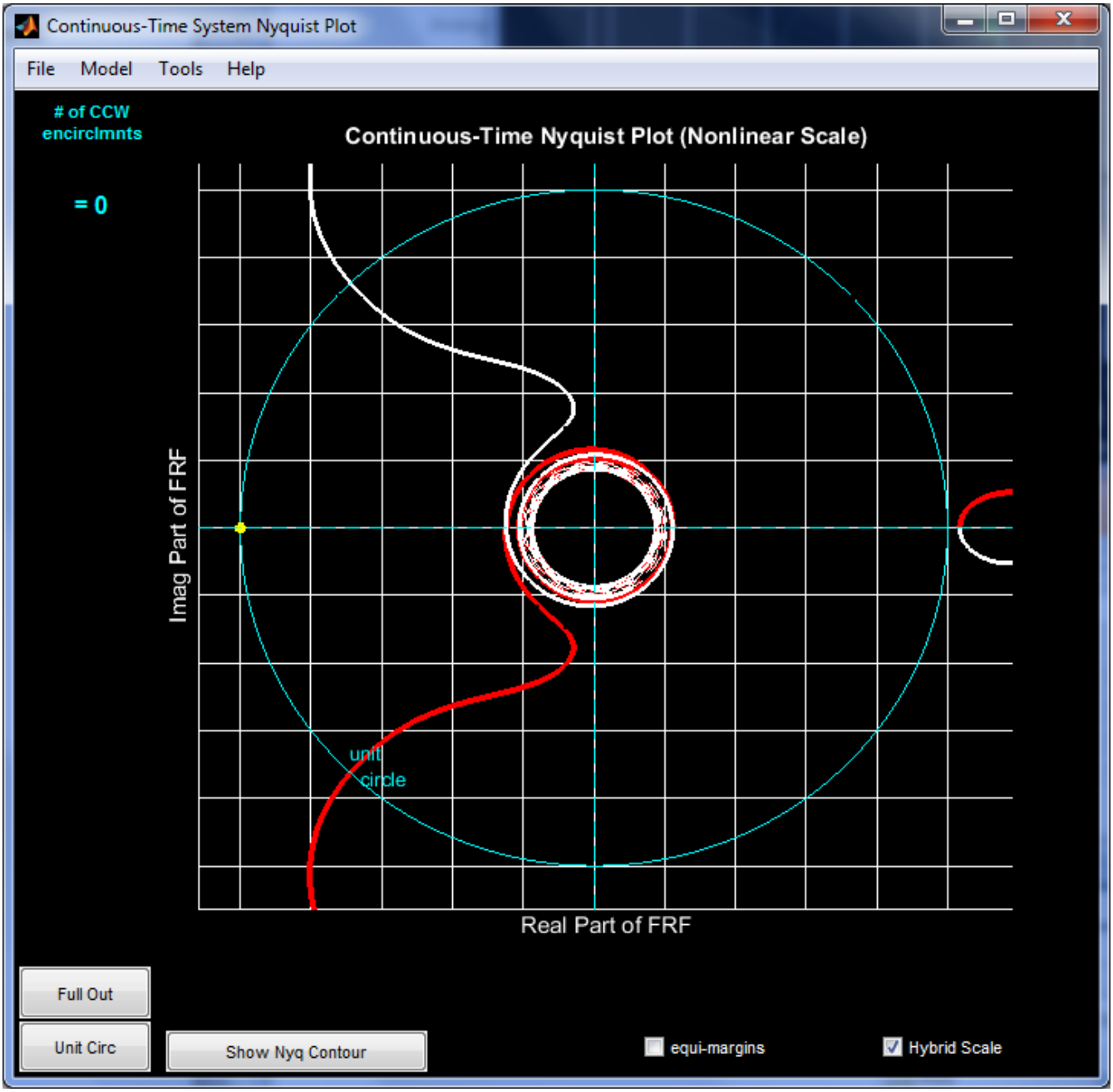
Because the Pade poles and zeros are so far from the transfer function poles and zero, they have very little effect on the root locus in the near vicinity of those transfer function poles and zero. The next plot has been "zoomed in" to a region close around the poles and zeros of the transfer function, to show that the effects of the relatively distant Pade poles and zeros are negligible.



By contrast, if the delay is changed from one millisecond to five milliseconds, the effect is dramatic. The root-locus in the near vicinity of the transfer-function poles and zeros is quite different. The tendency of delay to destabilize a system is clearly illustrated by this example.



Some interesting aspects of the *effects* of pure delay can be studied in PZGui. For example, through the use of nonlinear scaling (the "**Hybrid Scaling**" checkbox), the Nyquist plot can show a different perspective on the frequency-response effects of delay. As shown in the following figure, the Nyquist plot shows how at high frequencies the frequency response function "spirals in" to zero.



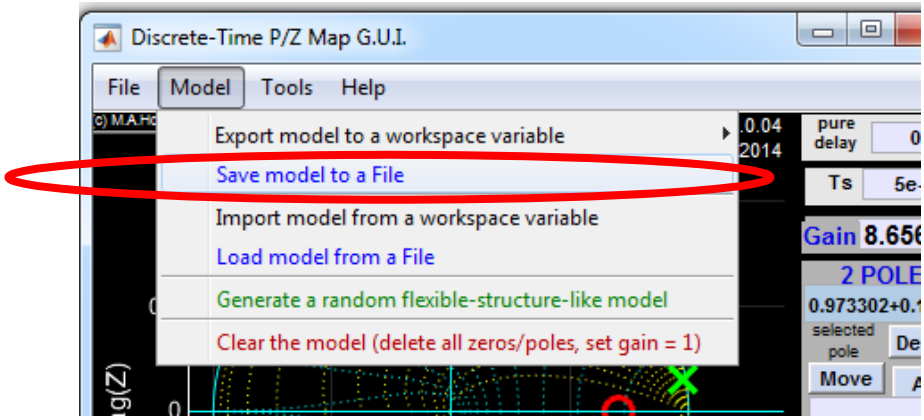
XIII. Saving Models and Exporting Models

Transfer function models that are in **PZGui** can be saved in two ways.

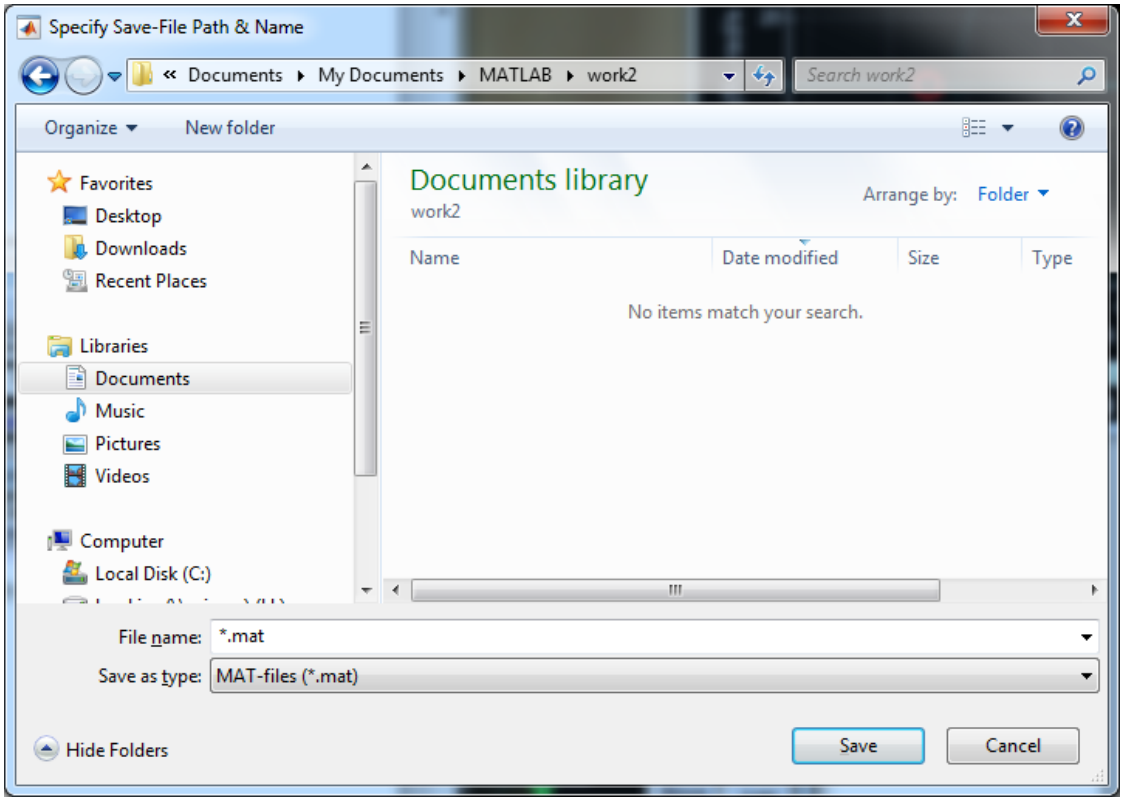
(1) Save to a file

(2) Export to a Matlab workspace variable

(1) Save to a file



This is fairly self-explanatory. The menu selection highlighted immediately above brings up the standard Matlab save-file user-interface, as seen in this example.

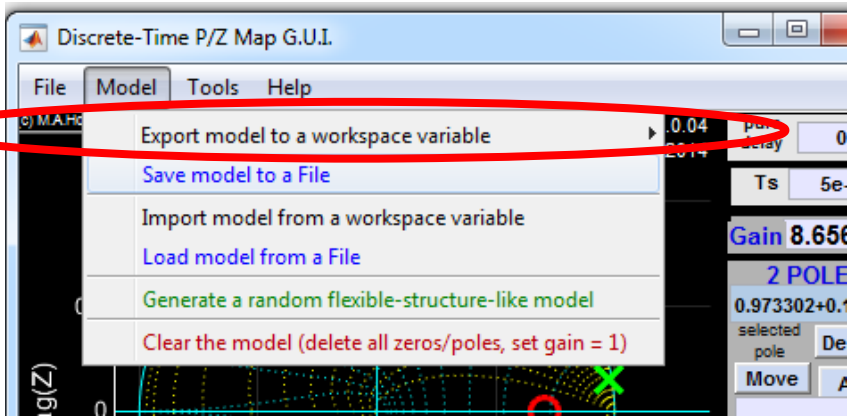


When the model is saved in this way, **both** the *continuous-time* model and the *discrete-time* model currently in **PZGui** are saved. Later, if the saved file is loaded, there is an option to load either one or both of the saved models.

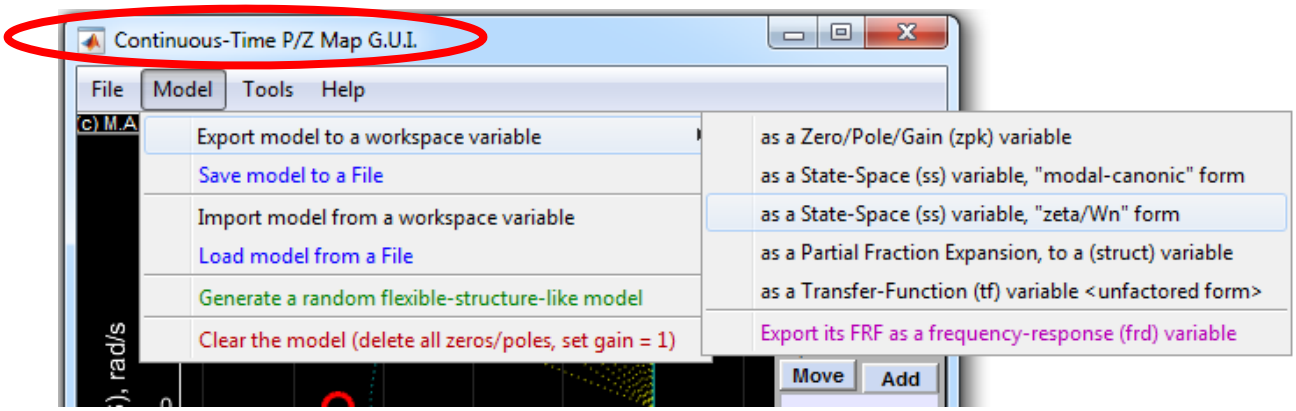
(2) Export to a Matlab workspace variable

You can export the current **PZGui** model into a Matlab workspace variable using the menu selection highlighted below. If you select this option from a continuous-time figure, the continuous-

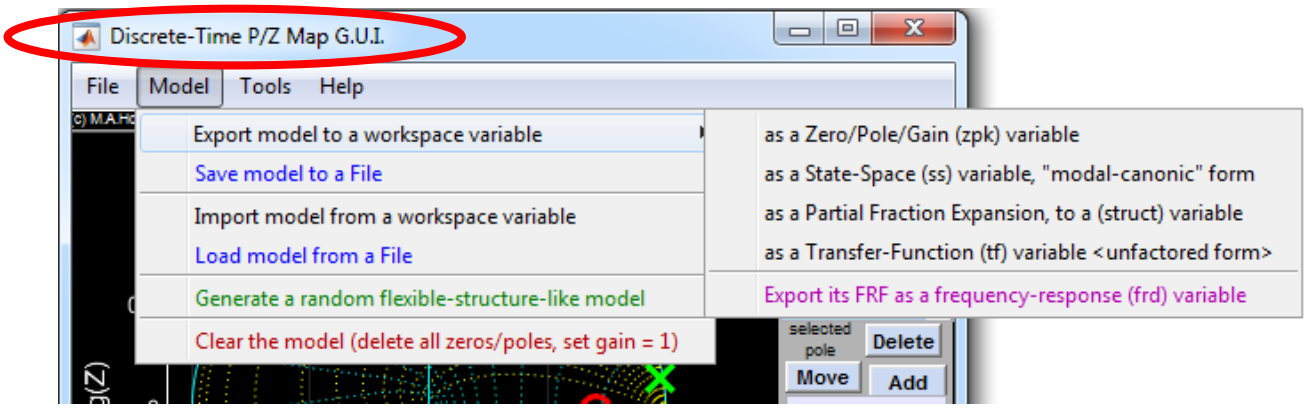
time model will be exported. Likewise, if you select this option from a discrete-time figure, the discrete-time model will be exported. You will be prompted to give a name to the variable.



This menu selection has a submenu, as shown here for the ***continuous-time*** case.



There is a slight difference between the submenus for the continuous-time and discrete-time models. Specifically, a discrete-time model ***cannot*** be saved in "zeta/Wn" form. (There is more discussion about the various forms of model, below.) Here is the ***discrete-time*** submenu, which does not offer that option.



Let us consider the various submodel selections, one-by-one.

(a) Zero/pole/gain (zpk) form

The **`PZGui`** model is exported as a Matlab Controls Toolbox **zpk**-object. Mathematically, this is a very well-conditioned form, because all pole locations and all zero locations are specified exactly. For more information about Matlab's **zpk**-object, at the Matlab command line, type

```
>> doc zpk
```

(b) State-space (ss) modal-canonic form

The modal-canonic form of a state-space model has the mathematically best-conditioned *A*-matrix of all real-valued similarity transformations. That's because the eigenvalues can be determined exactly, with no computation required.

Here is an example of modal-canonic form. Suppose a model has the following poles**:**

$$[\ (-3+31i\ )\quad (-3-31i)\quad (-20)\ ].$$

A modal-canonic *A*-matrix for a state-space model of this system is any one of the following.

$$\begin{bmatrix} -3 & -31 & 0 \\ 31 & -3 & 0 \\ 0 & 0 & -20 \end{bmatrix},\quad \begin{bmatrix} -3 & 31 & 0 \\ -31 & -3 & 0 \\ 0 & 0 & -20 \end{bmatrix},\quad \begin{bmatrix} -20 & 0 & 0 \\ 0 & -3 & -31 \\ 0 & 31 & -3 \end{bmatrix},\quad \begin{bmatrix} -20 & 0 & 0 \\ 0 & -3 & 31 \\ 0 & -31 & -3 \end{bmatrix}$$

More generally, the modal-canonic *A*-matrix is block-diagonal, with a 1x1 block for each real-valued eigenvalue, and a 2x2 block for each complex-conjugate pair of eigenvalues. In each 2x2 block, the two diagonal elements are the real parts, and the two off-diagonal elements are the imaginary parts, of the eigenvalues. Thus, the eigenvalues can be determined "by inspection", without any computation.

When there are pole-repetitions, the modal-canonic form is not as "pure". For example, here is a modal-canonic *A*-matrix when a triply-repeated pole at **-5** is appended to the previous example (assuming complete observability and controllability)**:**

$$\begin{bmatrix} -3 & -31 & 0 & 0 & 0 & 0 \\ 31 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & -20 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & 1 & 0 \\ 0 & 0 & 0 & 0 & -5 & 1 \\ 0 & 0 & 0 & 0 & 0 & -5 \end{bmatrix}$$

The matrix has a 3x3 Jordan-block for the eigenvalues at **-5**.

It gets yet more complicated if there are non-real-valued repeated-poles, and one could argue that the resulting *A*-matrix is not actually in modal-canonic form. Whatever the classification, the resulting form is still quite useful, and extremely well-conditioned (again, because the eigenvalues can be determined exactly, without the need for any computation).

When you elect to export the model to modal-canonic form, all these details are handled correctly, even when there are repeated poles.

For more information about Matlab's **ss**-object, at the Matlab command line, type

```
>> doc ss
```

(c) State-space (ss) zeta/ω_n form

This option is only available in the export of continuous-time models.

In this form, the state-space model *A*-matrix is block-diagonal, with a 1x1 block for each real-valued eigenvalue, and a 2x2 block for each complex-conjugate pair of eigenvalues. Each 2x2 block, corresponding to eigenvalues $\sigma \pm i\omega$, has the form**:**

$$\begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}$$

where $\omega_n = |\sigma + i\omega| = |\sigma - i\omega|$ is known as the ***natural frequency***, and $\zeta = \frac{\sigma}{\omega_n}$ is known as the ***damping factor***.

(d) Partial-fraction expansion (PFE) form

The best-conditioned form of the transfer-function is the partial-fraction expansion form. In this form, the transfer-function is expanded into a sum of terms that have constants in the numerators, and denominators that depend upon the pole-repetition. For non-repeated poles, the denominators are first-order. For a repeated pole with repetition *m*, there will be *m* terms with denominators of increasing order, from 1 to *m*.

For example, suppose a model has the following **zpk**-form transfer-function**:**

$$H(s) = \frac{0.1\,(s+70)\,(s+20-20i)\,(s+20+20i)}{(s+3-31i)\,(s+3+31i)\,(s+2)^3}.$$

The partial-fraction expansion is

$$H(s) = \left\{ \frac{(2.41+3.61i)\times 10^{-3}}{s+3-31i} + \frac{(2.41-3.61i)\times 10^{-3}}{s+3+31i} \;\cdots \right.$$
$$\left. + \frac{4.83\times 10^{-3}}{s+2} + \frac{0.319}{(s+2)^2} + \frac{5.12}{(s+2)^3} \;+\; 0 \right\}$$

Notice that the constant *direct-term* is zero in this example. It will only be nonzero if the order of the numerator polynomial is the same as the order of the denominator polynomial.

There is no **pfe**-object defined in Matlab, so when you select this form, it is exported as a ***structure-variable*** with fieldnames that are similar to terms used in the documentation of the Matlab **`residue`** function (which can be used to find the partial-fraction expansion of a transfer function). The fieldnames are **residues**, **poles**, and **direct**.

In the preceding example, the export variable was named **my_pfe**, and it has the following fields**:**

$$\textbf{my\_pfe.residues} = \begin{bmatrix} (2.41+3.61i)\times 10^{-3} \\ (2.41-3.61i)\times 10^{-3} \\ 4.83\times 10^{-3} \\ 0.319 \\ 5.12 \end{bmatrix}, \quad \textbf{my\_pfe.poles} = \begin{bmatrix} -3+31i \\ -3-31i \\ -2 \\ -2 \\ -2 \end{bmatrix}, \text{ and}$$

$$\textbf{my\_pfe.direct} = 0.$$

Notice that when a pole is repeated, the residues (*i.e.*, numerators) are given in order of increasing denominator-order. That's the same way Matlab's **`residue`** function handles repeated poles.

(e) Transfer-function (tf) form

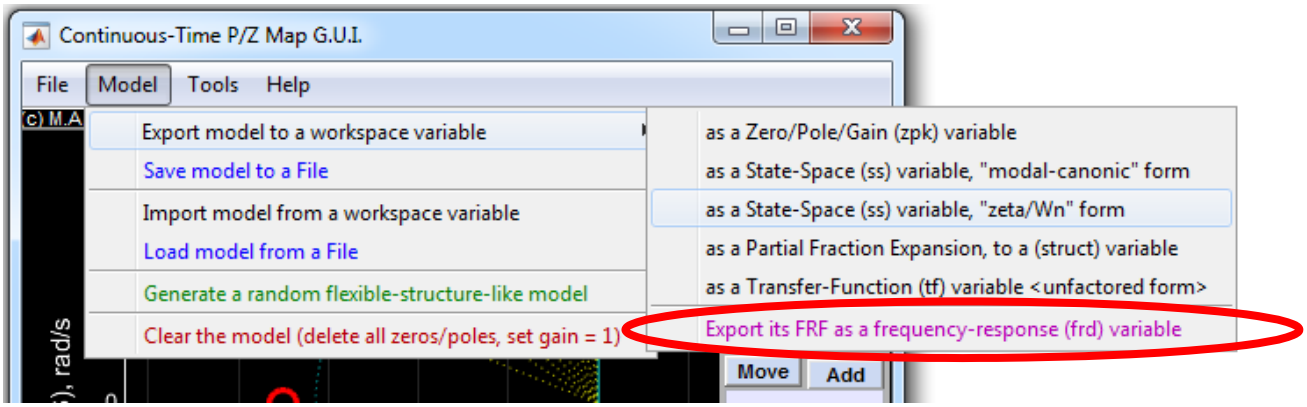
The worst-conditioned form of the transfer-function is as a ratio of two unfactored polynomials. This form is worst-conditioned because the poles and zeros can only be determined by using a root-finding routine. It's not too bad when model-order is less than about 12, but finding the roots of polynomials above that order is very difficult, and generally inexact. This is particularly true if there are any repeated poles or zeros, even when model-order is less than 12.

That said, the transfer-function form is supported by Matlab as an object-type, the **tf**-object. For more information about Matlab's **tf**-object, at the Matlab command line, type

```
>> doc tf
```

(f) Frequency-response data (frd) form

Although this method of exporting the model does not preserve any direct information about the poles, zeros, and gain of the transfer function, nevertheless it can be useful to save the frequency-response function of the model. That is the only purpose of this menu selection.



Be aware that the exported frequency-vector does not consist of equally-spaced frequencies. It generally doesn't consist of log-space frequencies, either, because **PZGui** uses an adaptive frequency-selection algorithm to "fill-in" more frequencies wherever magnitude and/or phase are changing quickly. The purpose of that in-filling is to guarantee smooth-looking frequency-response plots.

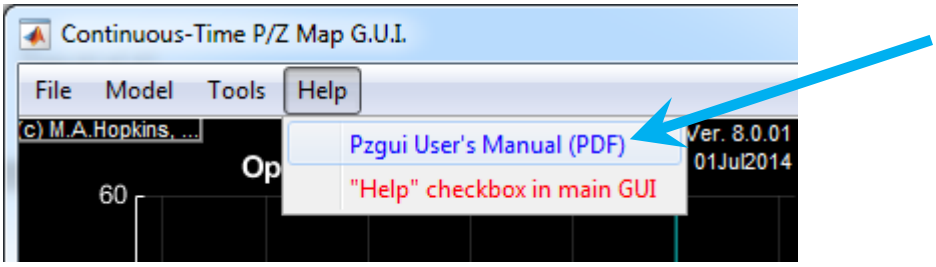
For more information about Matlab's **frd**-object, at the Matlab command line, type

```
>> doc frd
```

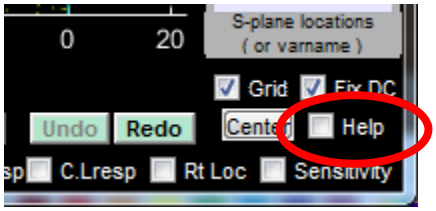
XIV. The "Help" Menu, and Context-Sensitive Help

There are only two ways to get help in **`PZGui`**.

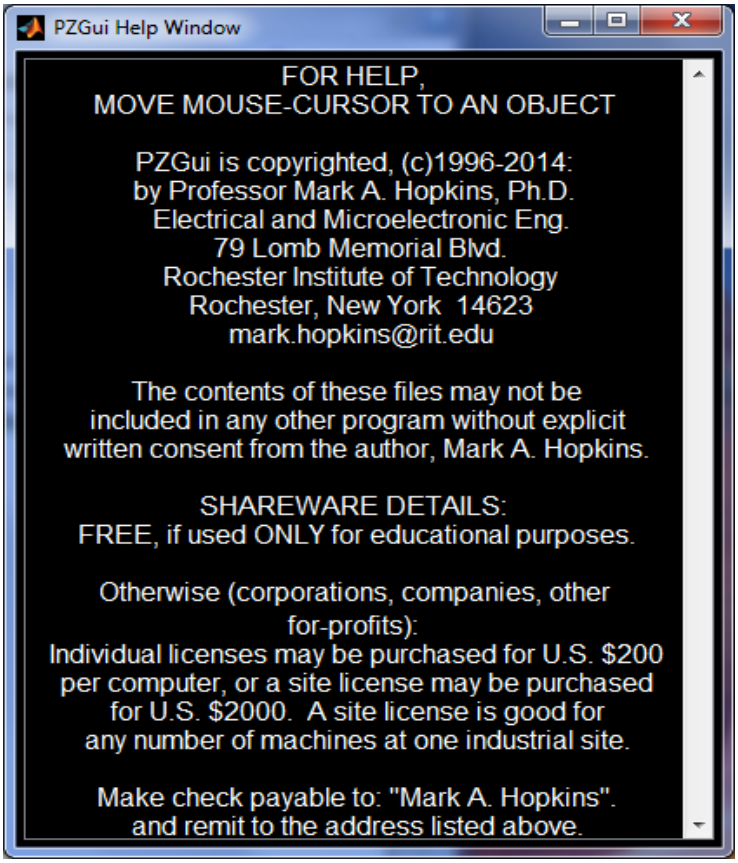
First, there is this self-same document, *User's Manual for `PZGui`, v.*8.0.xx, which can be accessed from the "**Help**" menu, provided that you have a PDF reader enabled.



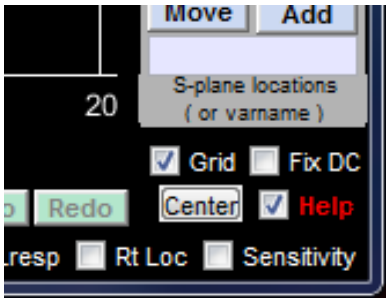
Second, there is a *context-sensitive* help that works only in the main pole/zero GUI. This can be activated by either of two methods: select the second menu item in the previous figure, "**"Help" checkbox in main GUI**", or click on the "**Help**" checkbox near the lower-right corner of the main pole/zero GUI, as shown in the next figure.



When the "**Help**" checkbox is selected, the following figure appears.



When the "**Help**" checkbox is selected, it is highlighted with red text, as follows.



If you close the Help Window figure, the "**Help**" checkbox will be de-selected.

Under certain circumstances, the "**Help**" checkbox will automatically be de-selected. For example, if you drag-and-drop a pole or zero, the help checkbox will be de-selected. It will also be deselected if you select "**Draw Box**" (Conversely, Draw Box is deselected when you select Help).

As you move the mouse-cursor around the main pole/zero interface, the text that appears in this figure changes, so that it always describes the object nearest to the mouse-cursor.

For example, if the cursor is placed near the "**OLbode**" checkbox (the pushbutton that creates the **open-loop Bode plot**), the Help Window text is updated as follows.